DETECÇÃO DE BORDAS E ANÁLISE DE TEXTURAS APLICADAS À INSPECÇÃO DE DUTOS SUBMERSOS

Filipe de Almeida Araujo Vital

Projeto de Graduação apresentado ao Curso de Engenharia de Controle e Automação da Escola Politécnica, Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Engenheiro.

Orientadores:
Profa. Mariane Rembold Petraglia, Ph.D.
Prof. José Gabriel R. C. Gomes, Ph.D.

Rio de Janeiro, RJ - Brasil
Março de 2014

DETECÇÃO DE BORDAS E ANÁLISE DE TEXTURAS APLICADAS À INSPECÇÃO DE DUTOS SUBMERSOS

Filipe de Almeida Araujo Vital

PROJETO SUBMETIDO AO CORPO DOCENTE DO CURSO DE ENGENHARIA DE CONTROLE E AUTOMAÇÃO DA ESCOLA POLITÉCNICA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO, COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA OBTENÇÃO DO GRAU DE ENGENHEIRO DE CONTROLE E AUTOMAÇÃO.

Examinado por:

Prof. José Gabriel Rodriguez Carneiro Gomes, Ph.D.
(Orientador)

Prof. Heraldo Luís Silveira de Almeida, D.Sc.

Prof. Diego Barreto Haddad, D.Sc.

Rio de Janeiro, RJ - Brasil
Março de 2014

Agradecimentos

Primeiramente, aos meus pais pela educação que recebi, sem ela não teria chegado onde cheguei.

Aos meus amigos, que me acompanharam e deram forças ao longo de toda a minha formação.

Aos meus orientadores Mariane Petraglia e José Gabriel Gomes pelos valiosos conselhos, paciência e apoio, tanto no desenvolvimento deste projeto de graduação como na continuidade da minha carreira acadêmica.

Finalmente, a todos os meus professores da graduação pelo conhecimento e experiência compartilhados.

Muito obrigado!

Resumo do Projeto de Graduação apresentado à Escola Politécnica da UFRJ como parte dos requisitos necessários para a obtenção do grau de Engenheiro de Controle e Automação.

Detecção de bordas e análise de texturas aplicadas à inspeção de dutos submersos

Filipe de Almeida Araujo Vital

Março de 2014

Orientadores: Profa. Mariane Rembold Petraglia e Prof. José Gabriel Rodriguez Carneiro Gomes

Curso: Engenharia de Controle e Automação

A verificação da integridade de dutos é muito importante na indústria de petróleo. Falhas no sistema de transporte e extração podem causar grandes impactos financeiros à empresa, tanto por perda do produto como por multas devido ao impacto ambiental.

Este trabalho propõe um método para utilizar técnicas de processamento de imagem para auxiliar os operadores nas inspeções remotas, tornando-as mais rápidas e confiáveis. As empresas responsáveis se beneficiariam tanto com a redução das horas de trabalho dos avaliadores como com a redução do risco de falhas no sistema, considerando, é claro, que os problemas encontrados fossem devidamente tratados.

É utilizada uma variação do algoritmo de Canny para detecção de bordas em imagens coloridas e a transformada de Hough para determinar a posição do duto, seguido de uma análise de textura utilizando DCT para buscar anomalias no duto.

Palavras-chave: Detecção de Bordas, Transformada Discreta de Cosseno, Inspeção Remota, Análise de Textura.

Abstract of Final Project presented to Poli/UFRJ as a partial fulfillment of the requirements for the degree of Engineer.

Edge detection and texture analysis applied to underwater pipelines inspection

Filipe de Almeida Araujo Vital

March 2014

Advisors: Prof. Mariane Rembold Petraglia and Prof. José Gabriel Rodriguez Carneiro Gomes

Specialty: Control and Automation Engineering

The inspection of pipeline integrity is very important in the oil industry. Failures in the transport and extraction system can cause large financial impact on the company, either by loss of the product or as fines due to environmental impact.

This work proposes a method of using image processing to assist operators in remote inspections, making them faster and more reliable. Responsible companies would benefit from both the reduction of working hours of the operators and the reduction of the risk of system failures, considering, of course, that the problems encoutered are properly treated.

A variation of the Canny algorithm for edge detection in color images and the Hough transform are used to determine the position of the pipeline, followed by texture analysis using DCT to find anomalies on the pipeline.

Keywords: Edge Detection, Discrete Cosine Transform, Remote Inspection, Texture Analysis.

Sumário

Lista de Figuras

1 Introdução

O presente trabalho de conclusão de curso apresentará um método para realizar a inspeção semi-automática de dutos submersos por meio de técnicas de processamento digital de imagens, notadamente detecção de bordas e análise de textura, aplicadas a sinais de vídeo.

1.1 Contexto

Dutos têm grande importância na indústria de petróleo, tanto na extração como no transporte. Devido ao grande impacto financeiro e ambiental causado por danos nesses dutos é muito importante um acompanhamento do seu estado de integridade. Parte desses dutos se encontra submersa, o que dificulta a sua inspeção sem equipamento apropriado.

Atualmente essa inspeção é feita por meio de vídeos filmados por ROVs (*Remotely Operated Vehicles*). Esses vídeos são analisados por operadores humanos, que verificam o estado do duto, identificam os possíveis problemas e geram um relatório discriminando os problemas encontrados e suas localizações. No entanto, como os dutos são muito extensos, os vídeos gerados são muito longos e sua análise pode se tornar tediosa. A diminuição da atenção dos operadores durante a análise pode diminuir a qualidade dela, aumentando o risco de que problemas no duto passem despercebidos.

Assim como os ROVs são utilizados para aumentar a segurança e diminuir o custo devido à mão-de-obra, o ideal seria poder substituir os operadores que analisam o vídeo por algum *software* especializado. A inspeção automática dos vídeos diminuiria o gasto com mão-de-obra e aumentaria a eficiência da inspeção.

1.2 Objetivo

O objetivo deste trabalho é propor um método para inspecionar o duto de forma semi-automática. O programa define a localização do duto no vídeo e busca anomalias nesta área pré-definida, indicando ao final os momentos do vídeo em que foram encontradas anomalias.

Esse programa não visa a substituição do operador, trata-se apenas de uma ferramenta que pode ser utilizada para auxiliar na inspeção, apontando os locais mais prováveis de anomalias no duto. Como não seria capaz de classificar as anomalias encontradas, o *software* ainda necessitaria de um operador para verificar se o evento encontrado é válido e classificá-lo.

1.3 Motivação

A motivação deste trabalho é evitar que o operador tenha que assistir ao vídeo completo da inspeção durante sua análise. Dessa forma ele poderá executar seu trabalho mais rapidamente, diminuindo o seu tempo de trabalho e os custos da empresa. Além disso, o operador não precisará se manter concentrado por longos períodos de tempo, diminuindo as chances de falha na inspeção devido à falta de atenção do operador.

1.4 Conteúdo do Trabalho

Este trabalho está dividido em três partes. A primeira parte é composta pelas Seções 1 e 2, que contêm esta introdução ao trabalho e uma revisão teórica das ferramentas que foram utilizadas. A segunda parte é composta pelas Seções 3, 4 e 5, e contêm o desenvolvimento do *software*. A terceira parte é composta pelas Seções 6 e 7, e contêm a apresentação dos resultados obtidos e a conclusão do trabalho.

2 Revisão Teórica

Nessa seção será apresentada uma breve revisão teórica a respeito das ferramentas utilizadas no desenvolvimento do projeto.

2.1 Redução de Ruído

A redução de ruído é o processo de remoção de grande parte do ruído de um sinal. Esse processo é muito importante no pré-processamento, pois não existe um dispositivo de aquisição perfeito. Todos possuem características que acabam introduzindo algum ruído no sinal captado. Além do ruído em geral não contribuir com informação, muitos algoritmos de processamento de sinais são sensíveis ao ruído. Logo, algoritmos de redução de ruídos são utilizados como pré-processamento visando melhorar o desempenho dos algoritmos que serão aplicados em seguida.

2.1.1 Filtros Gaussianos

O filtro gaussiano é um filtro cuja resposta ao impulso é uma função gaussiana (Figura 1). Como a transformada de Fourier de uma gaussiana é também uma gaussiana centrada em baixas frequências, esse filtro funciona como um filtro passa-baixa e é utilizado na suavização de imagens e redução de ruído [2].

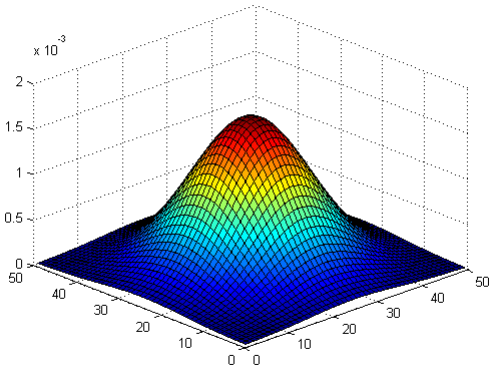
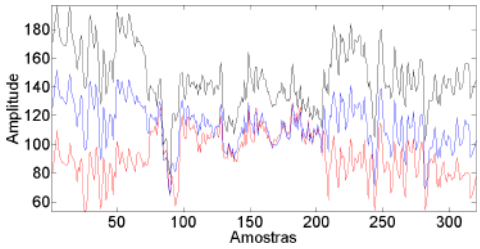
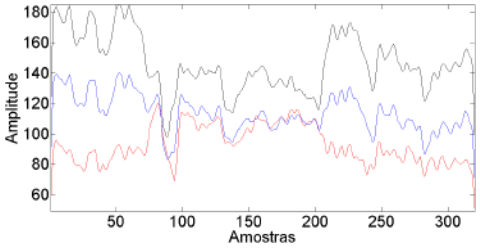


Figura 1: Filtro Gaussiano 2D

A maioria dos algoritmos de detecção de bordas são muito sensíveis ao ruído. A filtragem gaussiana é um dos métodos utilizados para reduzir o ruído e melhorar o desempenho da detecção. No entanto, deve-se lembrar que se o filtro não for bem ajustado pode causar um borramento excessivo



(a) Sinal antes da aplicação do filtro Gaussiano



(b) Sinal após a aplicação do fitro Gaussiano

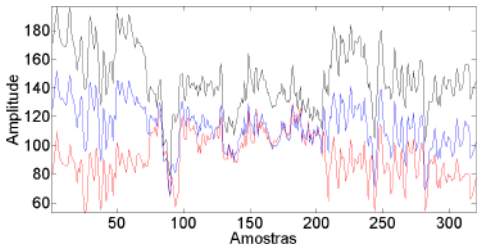
Figura 2: Efeito de um filtro Gaussiano de largura=5 e σ=1.5

da imagem e, consequentemente, das bordas, dificultando a detecção destas. O efeito do filtro pode ser visto na Figura 2.

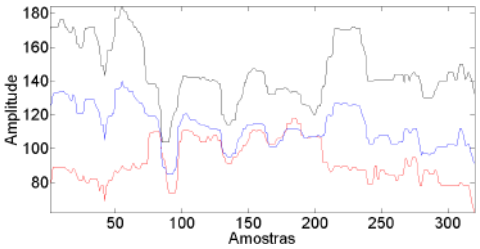
2.1.2 Filtro de Mediana

O filtro de mediana é um filtro não linear baseado na ordenação dos *pixels* contidos na área da imagem coberta pelo filtro. Ele substitui o *pixel* central pela mediana dos valores na vizinhança desse *pixel*. Ao ordenar um conjunto de valores, a mediana é o valor central [2].

O filtro de mediana, assim como o gaussiano, é muito usado para a redução de ruído, porém borra menos a imagem, impactando menos a nitidez das bordas. Um filtro de largura M elimina da imagem qualquer pulso de largura menor que $\left\lceil \frac{M}{2} \right\rceil$, sendo assim muito eficiente na rejeição de ruído impulsivo. Outra propriedade importante desse filtro é a de não afetar regiões constantes, bordas em degrau e bordas em rampa. O efeito do filtro pode ser visto na Figura 3.



(a) Sinal antes da aplicação do filtro de mediana



(b) Sinal após a aplicação do fitro de mediana

Figura 3: Efeito de um filtro de mediana de largura 11

2.2 Detecção de Bordas em Imagens em Níveis de Cinza

A detecção de bordas é uma das tarefas mais importantes em processamento de imagens e consiste em detectar descontinuidades significativas na imagem. Descontinuidades em imagens em geral correspondem a informações importantes, como descontinuidades na profundidade ou superfície, mudança de material ou variação na iluminação. Logo, é comum que essas técnicas sejam usadas como etapa inicial em outras técnicas de processamento, como segmentação de imagens, e detecção e rastreamento de objetos.

2.2.1 Modelos de Borda

Segundo González e Woods [2], os modelos de borda podem ser classificados de acordo com seus perfis de intensidade.

Borda em degrau A borda em degrau envolve uma transição entre dois níveis de intensidade que ocorre idealmente com uma distância de um *pixel*.

Borda em rampa Diferentemente da borda em degrau, a borda em rampa não tem um comportamento ideal. A transição entre os níveis ocorre em mais de um *pixel* e a nitidez da borda é diretamente proporcional à inclinação da rampa. Esse modelo se aproxima mais das bordas encontradas na prática, já que, devido às limitações dos equipamentos de aquisição, não conseguimos gerar imagens perfeitamente nítidas e livres de ruídos.

Borda em forma de telhado Esse tipo de borda é composto por uma rampa crescente seguida de uma rampa decrescente e é utilizada como modelo de linha. A largura da borda depende da nitidez e espessura da linha na imagem.

Devido ao ruído as imagens encontradas na prática dificilmente seguem perfeitamente esses modelos. No entanto, eles são muito úteis para definir matematicamente as bordas durante o desenvolvimento de algoritmos.

2.2.2 Detector de Bordas de Canny

O detector de bordas de Canny [1] é, atualmente, uma das melhores ferramentas para a detecção de bordas. Ele utiliza um algoritmo de várias etapas para identificar as bordas da imagem. O algoritmo consiste nas seguintes etapas:

1. Suavizar a imagem com um filtro gaussiano;

2. Calcular a magnitude e ângulo do gradiente da imagem;

3. Aplicar a supressão não máxima na imagem da magnitude do gradiente;

4. Usar a limiarização com histerese e análise de conectividade para detectar e conectar as bordas.

A saída gerada por esse algoritmo é uma imagem binária, na qual os *pixels* considerados bordas valem 1 e todos os outros valem 0. O resultado do algoritmo pode ser visto na Figura 4.

Suavização da imagem Como o gradiente é altamente afetado pela presença de ruído na imagem, um filtro Gaussiano é utilizado para suavizar a imagem e atenuar a influência do ruído.

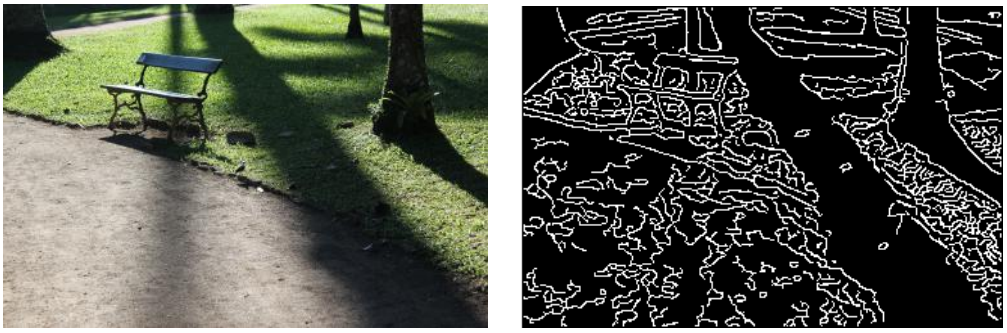
Essa etapa pode ser mesclada à etapa seguinte utilizando diretamente as derivadas vertical e horizontal de uma Gaussiana no lugar do filtro Gaussiano.

Cálculo do gradiente As bordas da imagem podem ter diversas orientações diferentes. Para ser capaz de detectar todas essas bordas, o algoritmo de Canny utiliza dois filtros para calcular os gradientes vertical e horizontal da imagem, g_v e g_h, respectivamente. A partir desses gradientes pode-se calcular a magnitude (M) e a direção (ϕ) do gradiente das bordas da imagem:

$$M = \sqrt{g_x^2 + g_y^2} \tag{1}$$

$$\phi = tg^{-1}\left(\frac{g_v}{g_h}\right) \tag{2}$$

Supressão não máxima As bordas encontradas na matriz M são largas, logo, para obter bordas mais finas é necessário que essa matriz passe por um algoritmo de afinamento. Na supressão não máxima são consideradas quatro direções de borda: horizontal, vertical, 45° e -45°. Para cada ponto $M(i, j)$, deve-se verificar qual é a direção mais próxima de $\phi(i, j)$ e se $M(i, j)$ é maior que seus vizinhos nessa direção. Em caso negativo, faz-se $M(i, j) = 0$, em caso positivo o valor de $M(i, j)$ não deve ser alterado. Com isso apenas serão consideradas como bordas os pontos máximos de cada região de borda detectada pelo gradiente.



(a) Imagem original

(b) Imagem gerada pelo algoritmo de Canny

Figura 4: Resultado do algoritmo de detecção de bordas de Canny

Limiarização Na etapa final do algoritmo é utilizada a limiarização por histerese. São definidos dois limiares: um alto e outro baixo. O limiar alto é utilizado para definir os pontos em que há mais confiança na veracidade da borda. Os pontos com valores acima desse limiar serão considerados bordas independente de seu posicionamento. O limiar baixo é utilizado para completar as bordas já encontradas. Os pontos cujos valores estão entre esses limiares serão considerados bordas apenas se forem vizinhos de um ponto de borda.

2.2.3 Transformada de Hough

Proposta por Paul Hough [3], esta técnica é utilizada para a identificação de linhas em imagens binárias por um processo de votação. Retas podem ser parametrizadas como:

$$\rho = x\cos(\theta) + y\sin(\theta) \tag{3}$$

Logo, pode-se associar cada linha na imagem a um ponto (ρ, θ) no espaço de parâmetros. Com base nessa parametrização são contados quantos pontos não-nulos da imagem original estão contidos em cada uma das possíveis linhas.

Inicialmente é definida a precisão desejada para os parâmetros ρ e θ e seus limites. Essa parametrização da reta é única para um intervalo de largura π em θ com $\rho \in \mathbb{R}$ e para um intervalo de 2π em θ com $\rho \geq 0$. Com essas informações é gerada uma matriz P, onde cada elemento $P(i, j)$ representará uma parte do espaço de parâmetros. As dimensões de P dependem da precisão escolhida inicialmente.

A matriz P é inicialmente preenchida com zeros. Então, para cada ponto

(x_i, y_i) não-nulo da imagem, varia-se o valor de θ de acordo com os intervalos permitidos e, utilizando a Eq.(3), calcula-se o ρ correspondente arredondando para o valor permitido mais próximo. Para cada par (ρ_j, θ_k) encontrado dessa forma, faz-se $P(j,k) = P(j,k) + 1$, onde $P(j,k)$ é o elemento da matriz que representa a parte do espaço de parâmetros que contém o par (ρ_j, θ_k).

Ao fim desse processo temos a matriz P preenchida com a quantidade de pontos da imagem que pertencem às retas possíveis. Os elementos de maiores valores indicam as retas mais representativas da imagem.

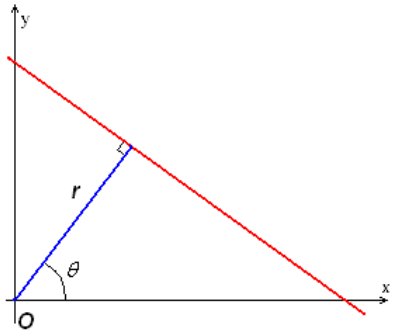
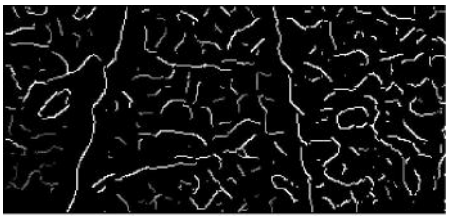
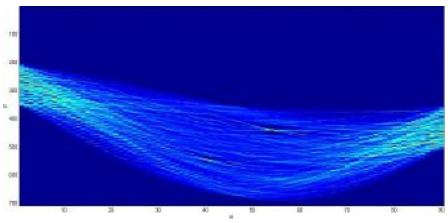


Figura 5: Parametrização usada na transformada de Hough.
Fonte: http://upload.wikimedia.org/wikipedia/en/e/e6/R_theta_line.GIF



(a) Imagem binária



(b) Matriz gerada pela aplicação da transformada de Hough nessa imagem

Figura 6: Exemplo do resultado da aplicação da transformada de Hough

2.3 Detecção de Bordas em Imagens Coloridas

O olho humano consegue discernir milhares de tons e intensidades diferentes de cores, no entanto só é capaz de identificar duas dúzias de tons de cinza. O processamento de imagens coloridas permite uma melhor análise das imagens pelo computador, aproximando-se mais do desempenho humano. Existe muita informação em imagens coloridas que é simplesmente ignorada

quando essa imagem é convertida para escala de cinza. Duas cores completamente diferentes podem ter a mesma representação em tons de cinza. Isso impossibilita a detecção de certas características na imagem devido à perda de informação.

A grande diferença entre imagens coloridas e em escala de cinza é que os *pixels* das imagens coloridas são representados por vetores, geralmente de três componentes, enquanto nas imagens em escala de cinza os *pixels* assumem valores escalares. Tendo isso em vista, duas abordagens são possíveis [5]:

- Utilizar técnicas monocromáticas, realizando as operações em cada componente separadamente e depois combinando os resultados obtidos.
- Utilizar técnicas vetoriais para tratar os *pixels* da imagem como vetores.

2.3.1 Detector de Canny para Imagens Coloridas

Uma das técnicas utilizadas na abordagem vetorial é uma adaptação do algoritmo de Canny, como pode ser visto na revisão feita por Koschan [5]. Sendo vetorial ela tira total proveito das informações contidas na imagem. Essa adaptação segue as mesmas etapas explicadas na Seção 2.2.2. Porém, como o gradiente utilizado no algoritmo original não é definido para vetores, foi necessário modificar essa etapa e definir outra forma de encontrar as bordas.

Considerando $\mathbf{f}$ a imagem colorida a ser processada, $\mathbf{f}_x$ e $\mathbf{f}_y$ são as derivadas espaciais da imagem e são dadas por

$$\mathbf{f}_x = \begin{bmatrix} R_x \\ G_x \\ B_x \end{bmatrix} \tag{4}$$

$$\mathbf{f}_y = \begin{bmatrix} R_y \\ G_y \\ B_y \end{bmatrix} \tag{5}$$

onde $R_x = \dfrac{\partial R}{\partial x}$ e $R_y = \dfrac{\partial R}{\partial y}$. O tensor de cor será definido como [5, 9]

$$\mathbf{G} = \begin{bmatrix} \mathbf{f}_x\mathbf{f}_x & \mathbf{f}_x\mathbf{f}_y \\ \mathbf{f}_x\mathbf{f}_y & \mathbf{f}_y\mathbf{f}_y \end{bmatrix} \tag{6}$$

A direção da maior variação na imagem é representada pelo autovetor correspondente ao maior autovalor de $\mathbf{G}$. A direção ϕ da borda pode ser

determinada por

$$\tan(2\phi) = \frac{2\mathbf{f}_x\mathbf{f}_y}{\|\mathbf{f}_x\|^2 - \|\mathbf{f}_y\|^2} \tag{7}$$

e sua intensidade m é indicada por

$$m^2 = \|\mathbf{f}_x\|^2 \cos^2(\phi) + 2\mathbf{f}_x\mathbf{f}_y \sin(\phi)\cos(\phi) + \|\mathbf{f}_y\|^2 \sin^2(\phi) \tag{8}$$

Sabendo o ângulo e a magnitude das bordas da imagem podemos aplicar as etapas 3 e 4 da Seção 2.2.2.

2.3.2 Gradiente Fotométrico Invariante

Visando melhor explorar as informações fotométricas das imagens, Weijer [8, 9, 10] propôs os gradientes fotométricos invariantes. Esses gradientes buscam utilizar as informações fotométricas para identificar algumas características físicas dos gradientes e, consequentemente, das bordas encontradas na imagem. Esses gradientes foram desenvolvidos a partir do modelo dicromático[7], que divide a reflexão em duas componentes: especular e difusa. Foi utilizada a hipótese de iluminação branca em toda a imagem e de uma refletância de interface neutra. De acordo com o modelo dicromático, o vetor RGB $\mathbf{f} = [R, G, B]$ pode ser definido como

$$\mathbf{f} = e(m^b\mathbf{c}^b + m^i\mathbf{c}^i) \tag{9}$$

onde $\mathbf{c}^b$ é a cor da refletância do corpo, $\mathbf{c}^i$ é a cor da refletância da interface, m^b e m^i são escalares representando a magnitude de reflexão, e e é a intensidade da fonte de luz. No caso de superfícies foscas não há reflexão de interface ($m^i = 0$), logo podemos simplificar o modelo para

$$\mathbf{f} = em^b\mathbf{c}^b. \tag{10}$$

Calculando a derivada espacial de (9) obtemos

$$\mathbf{f}_x = em^b\mathbf{c}^b_x + (e_x m^b + em^b_x)\mathbf{c}^b + (em^i_x + e_x m^i)\mathbf{c}^i. \tag{11}$$

Pode-se ver que a derivada espacial é composta pela soma de três vetores. Essas três parcelas representam, respectivamente, as influências da refletância do corpo, variações de sombreamento e variações especulares. Para obter os gradientes quasi-invariantes devemos subtrair do gradiente a projeção deste em um dos três vetores que o compõem.

Pode-se ver em (10) que para superfícies foscas o vetor $\mathbf{f}$ é paralelo à direção de variações de sombra $\mathbf{c}^b$. Logo, para superfícies foscas podemos definir o variante de sombra $\mathbf{S}_x$ e seu quasi-invariante $\mathbf{S}^c_x$ como

$$\mathbf{S}_x = (\mathbf{f}_x \cdot \hat{\mathbf{f}})\hat{\mathbf{f}} \tag{12}$$

$$\mathbf{S}_x^c = \mathbf{f}_x - \mathbf{S}_x \tag{13}$$

Para calcular o quasi-invariante especular e de sombra definimos uma direção $\hat{\mathbf{b}}$ perpendicular à direção de sombra $\hat{\mathbf{f}}$ e à direção da fonte de luz $\hat{\mathbf{c}}^i$:

$$\hat{\mathbf{b}} = \frac{\hat{\mathbf{f}} \times \hat{\mathbf{c}}^i}{\mid \mathbf{f} \times \mathbf{c}^i \mid}. \tag{14}$$

O quasi-invariante especular e de sombra $\mathbf{H}_x^c$ pode ser calculado projetando $\mathbf{f}_x$ em $\hat{\mathbf{b}}$, ou seja:

$$\mathbf{H}_x^c = (\mathbf{f}_x \cdot \hat{\mathbf{b}})\hat{\mathbf{b}} \tag{15}$$

Esses gradientes quasi-invariantes podem ser usados para substituir os gradientes comuns em (6) e aplicar o algoritmo de Canny desconsiderando bordas especulares e/ou de sombra, lembrando que seu desempenho depende da validade, na imagem em questão, das hipóteses feitas inicialmente.

2.4 Análise de Textura

Imagens reais são compostas por diversas texturas. Estas são facilmente identificáveis pelos seres humanos, porém são muito difíceis de serem definidas e ainda não existe uma definição precisa para elas. Pode-se dizer que a textura é o padrão de intensidades ou cores de uma região da imagem. Esses padrões são geralmente causados pelas características físicas do material na imagem, como rugosidade, geometria, refletância, granulosidade e cor. Logo, analisando a textura podemos também obter informações sobre as propriedades físicas dos diferentes objetos presentes na imagem.

As abordagens para análise de textura podem ser divididas em quatro categorias [6]:

Estrutural Estes métodos representam a textura por primitivas bem-definidas (microtexturas) e uma hierarquia de arranjos espaciais (macrotexturas) dessas primitivas.

Estatística Estes métodos tentam representar a textura indiretamente pelas propriedades não-determinísticas que governam as distribuições e relações entre as intensidades de uma imagem.

Baseada em modelo Estes métodos utilizam modelos estocásticos ou fractais para tentar interpretar a textura usando um modelo estocástico ou um modelo gerador de imagens. Os parâmetros do modelo são estimados e depois utilizados para análise de imagem.

No domínio da transformada Estes métodos utilizam transformadas como Fourier, Gabor, wavelet e DCT (transformada discreta de cosseno) para representar a imagem em um espaço cujas coordenadas tenham uma interpretação que é fortemente relacionada às características da textura, como, por exemplo, frequência.

Nesse trabalho foi utilizada apenas a abordagem de transformada, usando a DCT. Logo, nessa revisão teórica será abordada apenas a transformada de cosseno.

2.4.1 Transformada Discreta de Cosseno (DCT)

Assim como a transformada discreta de Fourier (DFT), a DCT expressa uma sequência finita de dados em uma base ortogonal, nesse caso funções cosseno de frequências variadas (Figura 7). A definição da DCT 2-D de uma sequência de comprimento N é

$$C(u,v) = \alpha(u)\alpha(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x) \cos\left[\frac{\pi(2x+1)u}{2N}\right] \cos\left[\frac{\pi(2y+1)v}{2N}\right], \quad (16)$$

para $u, v = 0, 1, 2, ..., N-1$ e

$$\alpha(u) = \begin{cases} \sqrt{\frac{1}{N}}, u = 0 \\ \sqrt{\frac{2}{N}}, u \neq 0 \end{cases} \quad (17)$$

Essa transformada é muito utilizada para compressão de dados devido à sua capacidade de concentrar a maior parte da informação da imagem em poucos coeficientes $C(u,v)$. Em geral para reconstruir uma imagem com um bom nível de precisão basta utilizar 50% dos pontos da DCT [4]. É essa mesma propriedade que faz ela ser interessante para aplicações que envolvem reconhecimento de padrões em imagens e análise de textura. A DCT tenta descorrelacionar os *pixels* da imagem, utilizando os padrões contidos na imagem para remover as redundâncias nos *pixels* vizinhos [4]. Esses padrões contidos nas imagens estão fortemente relacionados às texturas, logo pode-se considerar que os coeficientes da DCT têm uma forte influência das texturas, sendo muito úteis em sua análise.

Como pode-se ver na Figura 8, a informação da DCT está concentrada na parte superior esquerda da transformada. Essa concentração permite o descarte de grande parte dos *pixels* da DCT sem uma perda de informação significativa.

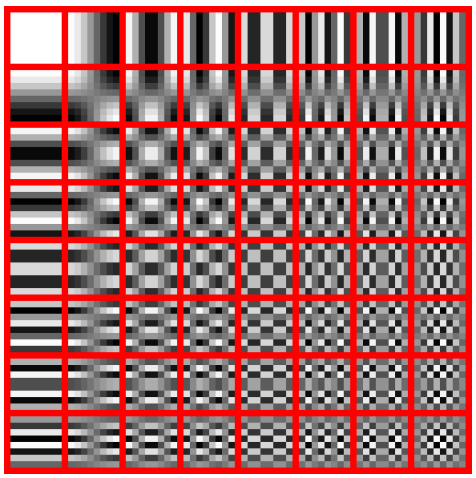


Figura 7: Funções base para uma DCT de dimensões 8×8.
Fonte: http://en.wikipedia.org/wiki/File:Dctjpeg.png

(a) Imagem original (b) DCT da imagem original

Figura 8: DCT aplicada a uma imagem real

3 Organização do Programa

Esta seção visa apresentar de forma sucinta o funcionamento geral do programa e as condições sobre as quais este foi desenvolvido.

3.1 Plataforma de Desenvolvimento

Este software foi desenvolvido totalmente em $MATLAB^{®}$, sendo utilizada a versão 2013a e os *toolboxes*:

- *Computer Vision System Toolbox*
- *Image Processing Toolbox*

Durante o desenvolvimento foi cogitada a migração para C++ e OpenCV para buscar uma maior eficiência do código. No entanto, devido às diferenças entre as versões de $MATLAB^{®}$ e OpenCV de certas funções, não foi possível utilizar as funções do OpenCV. Como as funções do $MATLAB^{®}$ se adequavam melhor ao método utilizado, o $MATLAB^{®}$ foi mantido como plataforma de desenvolvimento até o final do projeto.

3.2 Funcionamento do Programa

O programa pode ser dividido em 2 partes:

- Detecção de Dutos
- Análise de Textura

3.2.1 Detecção de Dutos

Como pode-se ver no diagrama de blocos da Figura 9, a deteção de dutos é a primeira operação a ser realizada. Esse bloco tem como objetivo definir os limites do duto em cada quadro do vídeo e calcular a DCT de uma janela dentro desses limites. Ele funciona como um *loop* em que cada iteração analisa um quadro do vídeo e parte da informação gerada é utilizada na iteração seguinte.

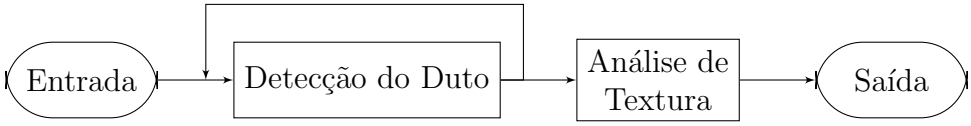


Figura 9: Diagrama de blocos do programa

Entrada Esse bloco recebe como entrada um vídeo em resolução 1280×720 filmado a 30 fps (*frames per second*). Esse vídeo contém uma filmagem de uma inspeção feita por um ROV.

Para fins de desenvolvimento foi considerado que a câmera se move a uma velocidade aproximadamente constante e foram utilizadas apenas as partes dos vídeos em que o duto estava completamente presente na imagem. Também foram desconsiderados vídeos nos quais o duto estava totalmente coberto de algas ou nos quais a água estava muito turva.

No desenvolvimento do *software* foram utilizados vídeos proprietários, por isso a exposição de imagens originais será limitada.

Saída Esta parte do programa gera um vídeo em resolução 320×150, no qual retas brancas marcam as posições encontradas das bordas do duto. Tanto a posição das bordas das retas como os parâmetros que as definem serão guardados para serem utilizados na próxima iteração do bloco.

Ao término desse bloco será passado para o bloco seguinte um vetor de matrizes 20×40 contendo parte dos coeficientes da DCT de uma janela localizada dentro dos limites encontrados para o duto. Cada elemento desse vetor é referente a um quadro do vídeo.

3.2.2 Análise de Textura

Este bloco tem como objetivo buscar anomalias nos dados fornecidos pelo bloco anterior. Ele se foca não em buscar um padrão específico pré-definido como dano, mas sim em buscar mudanças bruscas na textura do duto.

Entrada Este bloco recebe como entrada o vetor de matrizes 20×40 gerado pelo bloco anterior. Essas matrizes contém os coeficientes mais significativos das DCTs calculadas em cada um dos quadros do vídeo.

Saída Após realizar a análise, esse bloco fornece como saída um vetor contendo os quadros em que foram encontradas anomalias no duto.

4 Detecção do Duto

Nesta seção será detalhada a implementação da primeira parte do programa.

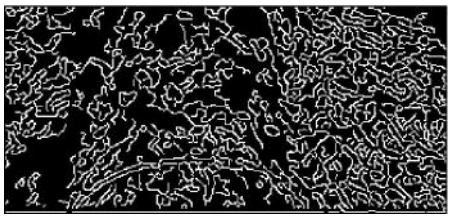
4.1 Pré-processamento

Como citado na Seção 3.2.1, o programa recebe como entrada um vídeo de uma inspeção em resolução 1280×720. Como a parte superior do vídeo contém um cabeçalho que atrapalharia o tratamento, esta é cortada, fazendo com que o tamanho real do vídeo a ser utilizado seja 1280×600.

O método utilizado para encontrar as bordas é computacionalmente pesado, logo realizá-lo na resolução total do vídeo seria muito demorado. Para contornar esse problema o vídeo é convertido para 320×150.

A Figura 10(a) contém um exemplo dos quadros que serão tratados pelo programa. Como pode-se ver o terreno ao redor do duto é bem heterogêneo. Isso torna a detecção do duto utilizando detecção de bordas um tanto quanto problemática. Mesmo que as bordas do duto possam ser encontradas utilizando o método de Canny, também são encontradas muitas bordas indesejadas. Podemos ver esse efeito na Figura 10(b), que foi gerada pela aplicação do algoritmo de Canny à versão monocromática da Figura 10(a).

(a) Quadro retirado de um dos vídeos utilizados no projeto



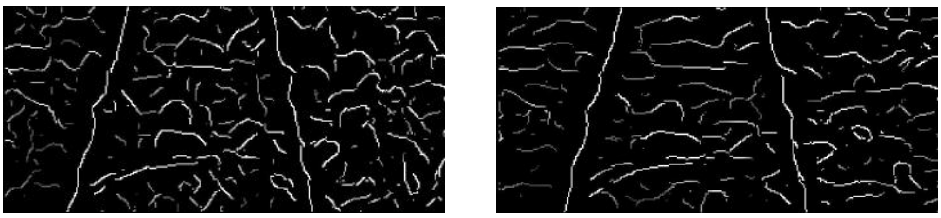
(b) Resultado do algoritmo Canny aplicado ao quadro ao lado

Figura 10: Exemplo da dificuldade em detectar o duto

Como foi mostrado na Figura 10, o algoritmo Canny para imagens em tons de cinza não foi capaz de encontrar as bordas do duto. Para tentar resolver esse problema foram utilizadas variações do Canny para imagens coloridas propostas por Van de Weijer [8, 9, 10]. Mais será dito a esse respeito na Seção 4.2. Embora o algoritmo para imagens coloridas tenha um desempenho satisfatório, muitas bordas indesejadas ainda são encontradas na imagem. Em geral essas bordas tem um direcionamento irregular e não se aproximam o suficiente de uma reta para confundir o algoritmo. No entanto, em alguns

casos alguns segmentos soltos podem possuir direções próximas e formar retas descontínuas que podem vir a causar erros na detecção.

Para tentar amenizar esse problema, foi aplicado um filtro de mediana. Graças a sua capacidade de eliminar pulsos de largura relativamente pequena, a aplicação de um filtro de mediana horizontal é capaz de eliminar a componente vertical de algumas bordas causadas pela heterogeneidade do terreno e/ou ruído. Como a borda do duto se assemelha a uma borda em degrau ou rampa, ela não é muito afetada pelo filtro. Pode-se ver na Figura 11 a diferença entre as bordas encontradas na imagem original e filtrada. Na imagem filtrada há uma quantidade menor de bordas verticais. A maioria tende para a direção horizontal. Como as bordas do duto estão sempre em direções próximas a vertical, essas bordas horizontais podem ser descartadas sem causar erros na detecção do duto. O impacto da filtragem em cada uma das componentes da imagem pode ser visto na Figura 3, que foi gerada a partir de uma linha horizontal na imagem. Cada um dos três sinais do gráfico representa uma componente dos *pixels* dessa linha da imagem.



(a) Bordas encontradas na imagem original

(b) Bordas encontradas na imagem filtrada

Figura 11: Comparação entre o resultado do algoritmo de Canny adaptado à imagens coloridas sem e com a filtragem de mediana

Após o ajuste de tamanho e resolução, e a aplicação do filtro de mediana horizontal, a imagem está pronta para ser utilizada na próxima etapa da detecção do duto.

4.2 Detecção de Bordas

O processo de detecção de bordas não funciona da mesma forma em todas as iterações do programa. Há um processo básico e a partir dele foram feitas algumas modificações para melhorar tanto o desempenho como a velocidade do algoritmo.

Na primeira iteração do programa é utilizado o processo básico, pois não há informação da iteração anterior. Em todas as outras iterações é utilizado o algoritmo aperfeiçoado, que utiliza informações da iteração anterior.

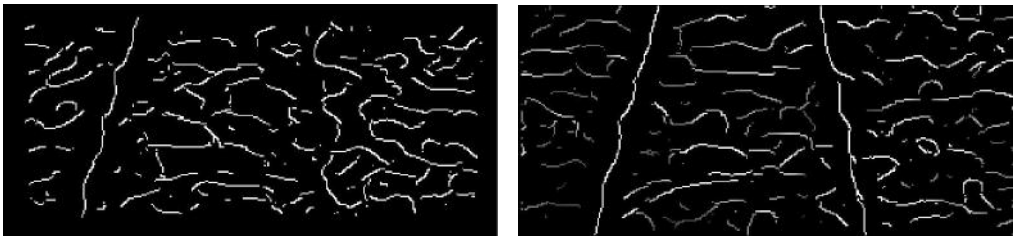
4.2.1 Processo Básico

Como mencionado anteriormente, o duto é detectado por meio de suas bordas. O algoritmo de detecção de bordas detecta as bordas do duto e todo o espaço entre essas bordas é considerado parte do duto. Na Seção 4.1 foi citada a utilização de uma variação para imagens coloridas do algoritmo de Canny para detecção de bordas. A explicação do funcionamento desses algoritmos pode ser vista na Seção 2.3. Foram citados na revisão teórica quatro tipos de algoritmo. O funcionamento deles é quase idêntico, tendo como diferença apenas o gradiente utilizado para encontrar as bordas da imagem. Um deles utiliza um gradiente comum e os outros utilizam gradientes fotométricos invariantes. Esses gradientes fotométricos invariantes foram propostos por Van de Weijer [8, 9, 10], com base no modelo dicromático de reflexão. Eles visam calcular um gradiente que seja invariante a variações específicas nas cores da imagem, como sombras e especularidades, de forma que as bordas causadas por esses tipos de variações não sejam detectadas.

Durante o desenvolvimento do projeto foram testados os quatro tipos de detector de bordas para imagens coloridas e seus resultados foram comparados para definir qual teria um melhor desempenho para essa tarefa. A Figura 12 mostra as bordas encontradas utilizando cada um dos quatro métodos. Podemos notar pelo exemplo que o método invariante à sombra (Figura 12(b)) e o invariante a especularidades (Figura 12(c)) têm desempenhos muito próximos e substancialmente melhores que todos os outros métodos. Após a realização de alguns testes foi constatado que o desempenho do método com gradiente invariante a especularidades possui um desempenho um pouco superior, principalmente na presença de algas no duto, por isso ele foi o escolhido para ser utilizado.

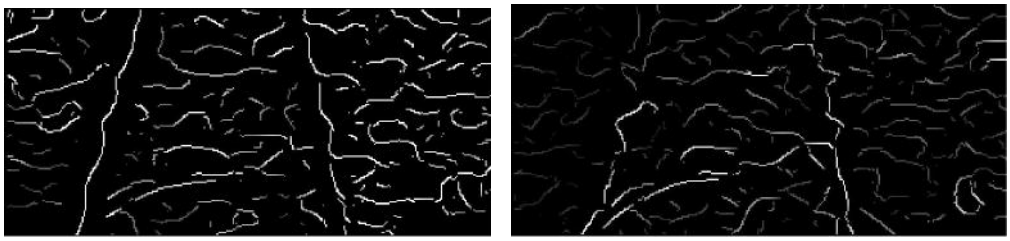
A aplicação do algoritmo de detecção de bordas gera uma imagem binária como as da Figura 12. Agora é necessário definir os contornos do duto a partir das bordas encontradas. Para tal, foi considerado que o duto não faz curvas, logo seus contornos podem ser aproximados por retas na imagem. Com isso em mente foi utilizada a transformada de Hough para buscar a reta que contém mais *pixels* de valor 1 nessa imagem. Como explicado na Seção 2.2.3, a transformada de Hough gera um espaço de parâmetros ρ e θ que podem definir qualquer reta possível na imagem e, através de um processo de contagem, verificar as retas que passam por mais *pixels* na imagem.

Se a detecção de bordas estiver funcionando suficientemente bem para detectar as bordas do duto e não houver outra borda, ou conjunto de bordas, na imagem que se aproxime de uma reta, as bordas do duto devem ser as duas retas mais bem representadas nessa imagem. Logo, elas seriam os dois máximos da matriz gerada pela transformada de Hough.



(a) Canny colorido com gradiente comum

(b) Canny colorido com gradiente invariante a sombra



(c) Canny colorido com gradiente invariante a especularidades

(d) Canny colorido com gradiente invariante a sombra e especularidades

Figura 12: Resultado das variações coloridas do algoritmo de Canny propostas por Van de Weijer. [8, 9, 10]

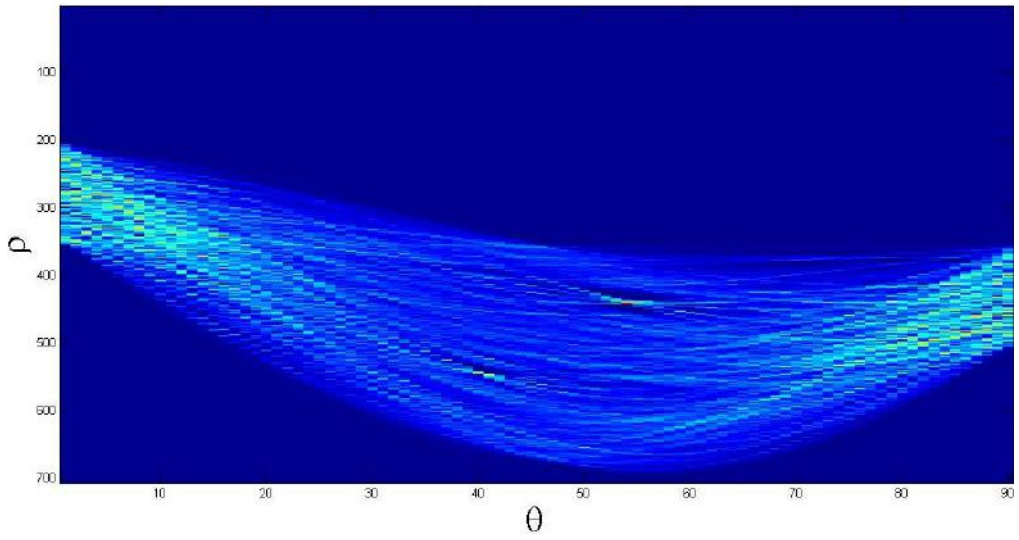


Figura 13: Resultado da aplicação da Transformada de Hough na Figura 12(c)

Na Figura 13 podem ser vistos dois máximos locais na parte central do gráfico e alguns máximos locais próximos às bordas do gráfico. Nesse gráfico

o eixo horizontal representa o valor de θ que varia de $-\frac{\pi}{2}$ a $\frac{\pi}{2}$. De acordo com a parametrização utilizada na transformada de Hough, que pode ser vista na Figura 5, uma reta aproximadamente vertical teria um θ aproximadamente igual a 0. Logo, como sabemos que na Figura 12(c) as bordas do duto estão em posição próxima à vertical, podemos deduzir que os dois máximos locais próximos ao centro são referentes às bordas do duto. Também podem ser vistos vários máximos locais para θ próximo de $-\frac{\pi}{2}$ e $\frac{\pi}{2}$, esses valores são referentes às retas próximas da horizontal. Para impedir que essas retas horizontais atrapalhem a detecção das bordas, foi considerado que o duto nunca está próximo da horizontal e as laterais da matriz de transformada são cortadas antes da busca pelos máximos nesta. Na Figura 14 pode ser visto resultado do corte na matriz de Hough e como isso faz com que haja apenas dois máximos locais significativos na matriz.

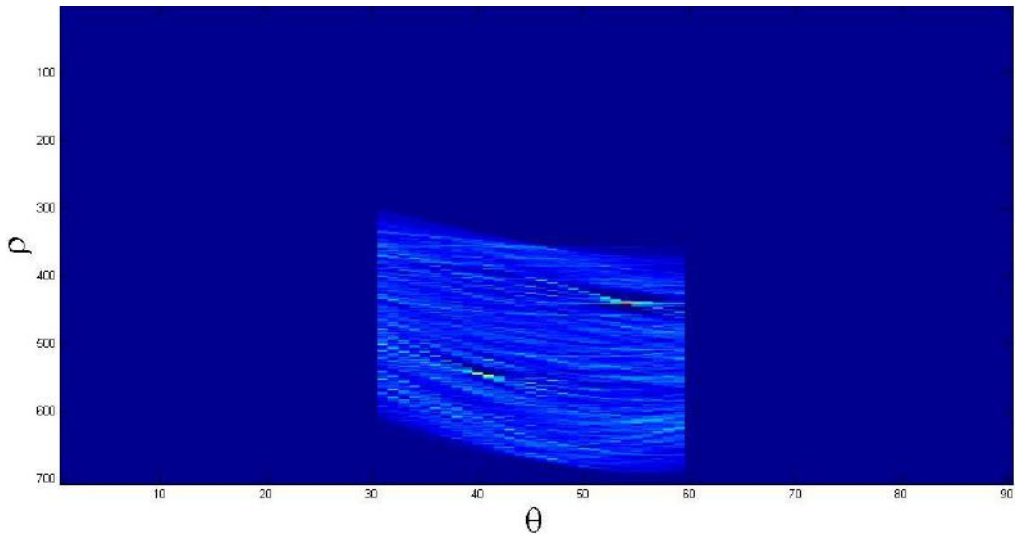


Figura 14: Matriz de Hough após o corte

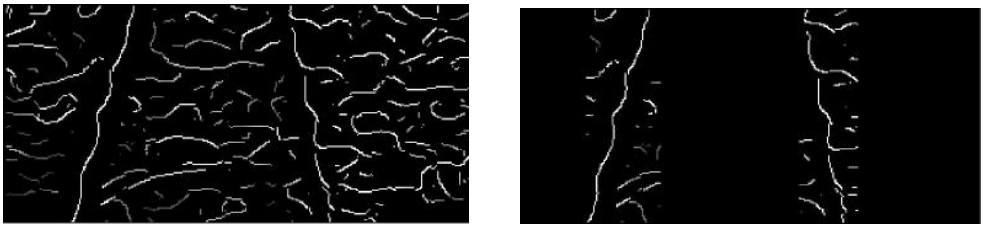
Após calcular os máximos da matriz, os valores de ρ e θ são utilizados para desenhar as retas no quadro (versão 320×150 sem filtragem) e é gerado um vídeo a partir dessas imagens. Os valores de ρ e θ, e as coordenadas das extremidades das retas são salvos para serem utilizados na iteração seguinte.

4.2.2 Processo Aperfeiçoado

A partir do segundo quadro do vídeo as informações das retas encontradas no quadro anterior podem ser utilizadas para melhorar o desempenho do método. O algoritmo funciona da mesma forma: primeiro as bordas são

calculadas, depois busca-se algo que se aproxime de uma reta e então essas retas encontradas são desenhadas na imagem. No entanto, são feitas algumas modificações para tentar evitar alguns possíveis problemas do método básico.

Um dos problemas visados nessa modificação é o custo computacional do algoritmo. Como o Canny para imagens coloridas considera um vetor para cada *pixel*, o seu custo computacional é muito superior ao da versão original, que considera apenas valores escalares. Mesmo reduzindo a resolução do vídeo o processo continua muito demorado. Como o vídeo foi filmado a 30 fps, o movimento do duto de um quadro para outro é muito pequeno. A partir do momento em que a posição do duto é conhecida em um quadro, pode-se afirmar que no quadro seguinte a localização e direção da bordas serão muito próximas às do quadro anterior. Logo, para diminuir o tempo gasto com o cálculo das bordas, o algoritmo de Canny é aplicado apenas na região próxima à borda encontrada no quadro anterior. A partir das extremidades das retas do quadro anterior é definida uma janela em torno de cada reta encontrada. Para definir a janela verifica-se qual extremidade da reta possui o maior valor de x, que nesse caso será chamado de x_1, e qual é o menor valor, que será chamado de x_2. Sendo δ um número inteiro a ser escolhido pelo usuário, a janela a ser utilizada é uma faixa vertical de $x_2 - \delta$ até $x_1 + \delta$. Quanto maior o δ, maior será o tamanho da janela. Com $\delta = 0$ a largura da janela será a distância horizontal entre as extremidades da reta do quadro anterior, lembrando que é calculada uma janela diferente para cada borda.



(a) Bordas calculadas pelo processo básico

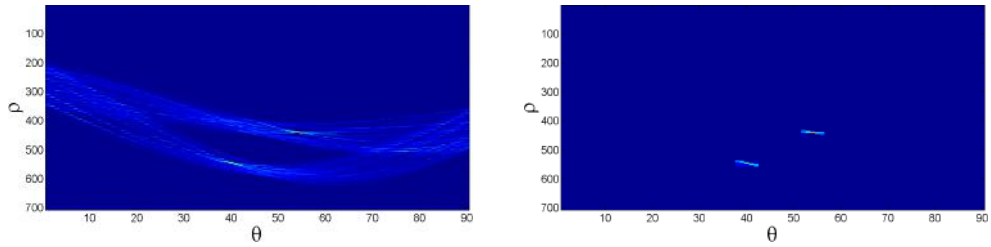
(b) Bordas calculadas pelo processo aperfeiçoado

Figura 15: Comparação entre as bordas que seriam calculadas utilizando o processo básico e o aperfeiçado

Pode ser visto claramente na Figura 15 que o método aperfeiçoado reduz substancialmente a região da imagem onde é necessário aplicar o algoritmo de Canny modificado. No caso mostrado a região a ser processada foi reduzida em aproximadamente 70%. Além da vantagem computacional, essa modificação ajuda a impedir que objetos na imagem influenciem na detecção do duto. Normalmente se houvesse um objeto com um contorno reto e bem

definido na imagem, ele poderia ser erroneamente identificado como uma borda do duto, o que atrapalharia a detecção. A aplicação local da detecção de bordas evita esse tipo de problema, pois objetos influenciarão na detecção apenas se estiverem muito próximos das bordas do duto.

Outro problema visado foi o de variações bruscas nas retas encontradas. Isso ocorre quando, por algum motivo, um dos dois máximos considerados da matriz de Hough passa a ter um valor menor do que o de um outro elemento distante da matriz. Isso pode ser causado por uma dificuldade na detecção da borda correta, seja por uma obstrução da borda ou por presença de algas, que fará com que o valor dessa reta na matriz diminua. Outro motivo seria caso um objeto com uma borda bem definida surgisse na imagem. Nesse caso o valor da reta equivalente a borda desse objeto aumentaria e poderia ultrapassar o valor de um dos máximos que representam os limites do duto. Para resolver esse problema foi utilizada uma abordagem semelhante à utilizada no primeiro problema. Ao invés de procurar pelos máximos em toda a matriz de Hough, será considerada apenas uma janela de tamanho definido pelo usuário em torno de cada máximo encontrado no quadro anterior. Essa modificação se baseia na mesma hipótese da modificação anterior, a de que o *framerate* do vídeo é suficientemente alto para que a variação das bordas do duto entre quaisquer dois quadros consecutivos seja muito pequena. O tamanho dessa janela deve ser calibrado de forma que seja suficientemente grande para incluir as variações que podem ocorrer entre os quadros, mas suficientemente pequena para diminuir a possibilidade de que uma reta errada seja encontrada.



(a) Matriz de Hough completa.

(b) Matriz de Hough considerada pelo algoritmo aperfeiçoado.

Figura 16: Comparação entre as matrizes de Hough antes e depois do corte.

Como pode ser visto na Figura 16, essa modificação reduz consideravelmente a quantidade de retas a serem consideradas na busca. Mesmo que um outro objeto apareça na imagem ou que o duto seja parcialmente coberto, a variação da reta será limitada pela janela e o algoritmo tem menos chances de indicar uma reta muito diferente da esperada.

4.3 Cálculo da DCT

Após calcular as bordas do duto ainda é necessário obter as informações de textura do duto para utilizar na próxima parte do programa. Nesta seção será descrita somente a forma como essas informações são geradas. Maiores explicações sobre o porque das decisões serão dadas na Seção 5 que trata especificamente da análise de textura.

Para o processamento de textura é utilizada uma versão em preto e branco do vídeo em resolução 1280×600, logo as informações referentes às bordas do duto devem ser convertidas para essa resolução. É utilizada uma janela de altura 160 e largura igual a 80% da largura do duto na borda superior da janela. Foi deixada uma distância entre a janela e os limites do duto para evitar que as bordas do duto e, possivelmente, o fundo influenciassem na análise de textura. Como a janela está localizada no terço inferior do duto e a altura da imagem é de 600 *pixels*, a borda superior da janela encontra-se no 200° *pixel*. A Figura 17 ilustra como seria essa janela. Os pontos p_1, p_2, p_3 e p_4 são as extremidades das retas encontradas pela transformada de Hough, devidamente convertidas para a nova resolução. Os demais pontos podem ser calculados a partir destes da seguinte forma:

$$p_5 = \frac{p_1 + 2p_2}{3} \tag{18}$$

$$p_6 = \frac{p_3 + 2p_4}{3} \tag{19}$$

$$p_7 = 0.9p_5 + 0.1p_6 \tag{20}$$

$$p_8 = 0.1p_5 + 0.9p_6 \tag{21}$$

Ao calcular a DCT desta janela obtém-se algo semelhante à Figura 18. É possível notar que a maior parte das informações da imagem estão contidas na parte superior da imagem. A DCT tem sempre o mesmo tamanho da janela que a gerou, como a janela possui tamanho variável o tamanho da DCT também varia. No exemplo mostrado na Figura 18 a janela é de 161×597. Como a maior parte da informação está contida em uma pequena parcela da matriz, foi decidido utilizar apenas um setor de tamanho constante no canto superior esquerdo da matriz. Além de diminuir a quantidade de informação a ser armazenada, uma matriz de tamanho constante também facilita o manuseio dos dados. Por meio de testes foi definido que uma janela de 20×40 no canto superior esquerdo, como na Figura 19 seria o suficiente para a análise de textura. Essas janelas 20×40 calculadas em cada quadro são salvas em um vetor de matrizes e repassados para a próxima parte do programa, a análise de textura.

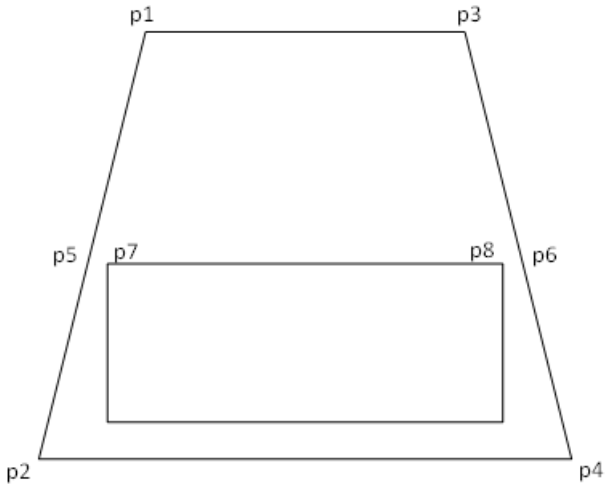


Figura 17: Esquema do posicionamento da janela a ser utilizada. O trapézio representa do duto na imagem e o retângulo representa a janela.

Figura 18: DCT de uma janela 161×597.

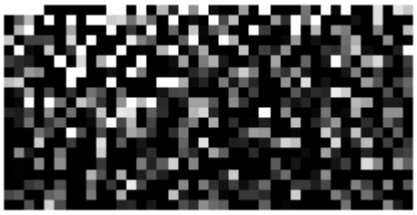


Figura 19: Parte 20×40 da matriz DCT que será armazenada para a análise de textura.

5 Análise de Textura

O objetivo desta seção é descrever o método utilizado para a análise de textura. Esta análise utiliza os dados obtidos na etapa anterior do programa para buscar anomalias no duto.

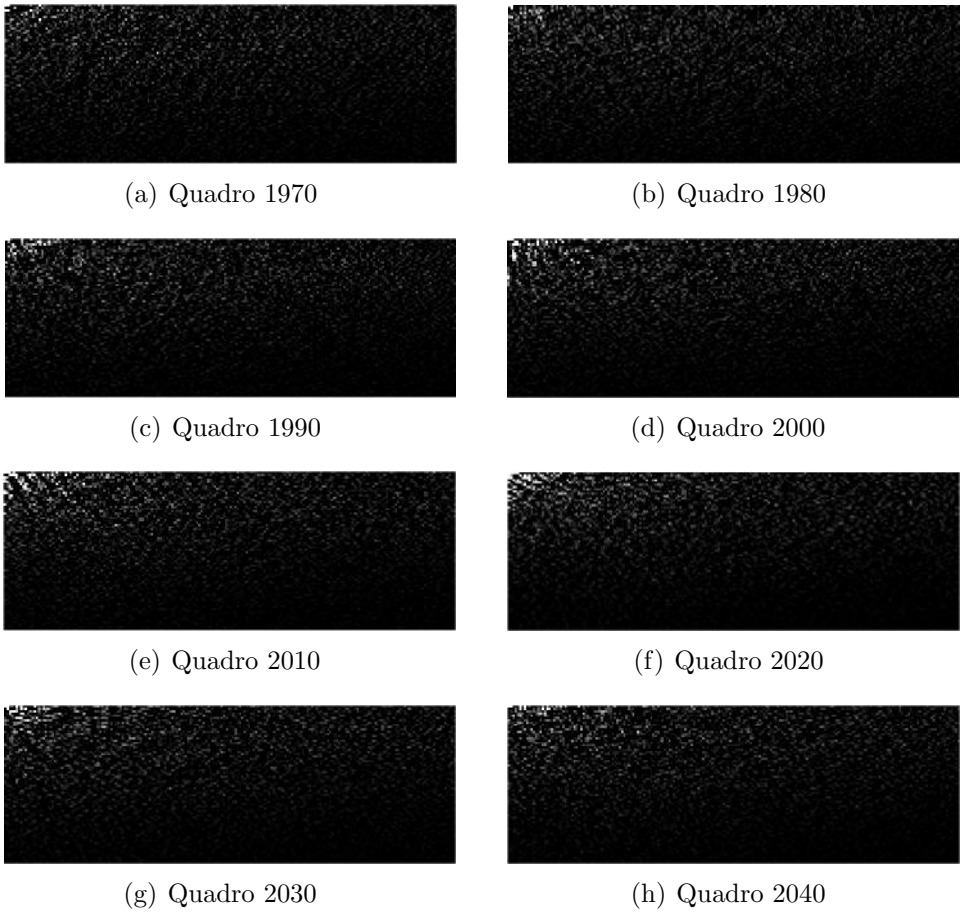
A análise de textura é feita por meio da análise dos coeficientes da DCT de uma janela definida na região do duto. Como a DCT é capaz de comprimir a informação da imagem em uma quantidade menor de pixels, pode ser feito um processamento mais eficiente dos dados. É esperado que regiões próximas do duto tenham aparência semelhante, seguindo padrões semelhantes de textura. Mesmo que ao longo do duto existam regiões que sejam claramente diferentes umas das outras devido aos diferentes ambientes aos quais o duto está sujeito, dentro de cada região o duto não deve ter variações bruscas. Logo, é esperado que os coeficientes da DCT das janelas retiradas do duto não possuam grande variação. Com base nisso esse algoritmo visa detectar variações inesperadas nos coeficientes da DCT, que podem ser causadas por anomalias no duto.

Para verificar o impacto desses eventos a serem detectados nos coeficientes da DCT, foram utilizados trechos de vídeos nos quais estavam presentes anomalias que deveriam ser detectadas. Esses testes foram realizados separadamente do algoritmo da detecção de bordas, logo foram utilizadas janelas de tamanho fixo. Foram escolhidas janelas que estivessem sempre dentro dos limites do duto no trecho considerado. Foram salvos os coeficientes de cada janela e foram comparados os trechos que não possuíam eventos com os que possuíam.

A Figura 20 mostra as DCTs relativas a uma sequência de quadros que contém um evento (quadro 1970 ao 2040). Os quadros foram impressos de 10 em 10. O tamanho da janela utilizada foi 157×445, porém, para facilitar a visualização, foi exibida apenas uma janela de 70×200 no canto superior esquerdo. Como pode ser visto no exemplo da Figura 18, o resto da DCT não contém informação relevante. Nessa sequência o evento começa a aparecer no quadro 1995 e some completamente da imagem no quadro 2035. Pode-se notar que a partir da Figura 20(d) o canto superior esquerdo da imagem fica mais claro.

Devido ao maior impacto dos eventos no canto superior direito da imagem, foi definido que uma janela de 20×40 no canto superior esquerdo da DCT seria utilizada na análise de textura. As janelas referentes às matrizes da Figura 20 podem ser vistas na Figura 21.

Como já dito anteriormente, o interesse desse algoritmo não é identificar texturas em especial, mas sim detectar uma variação inesperada na textura do duto. Por isso não será utilizada a matriz de DCT de cada quadro, mas sim a diferença entre a matriz de um quadro e do quadro anterior.



(a) Quadro 1970 (b) Quadro 1980

(c) Quadro 1990 (d) Quadro 2000

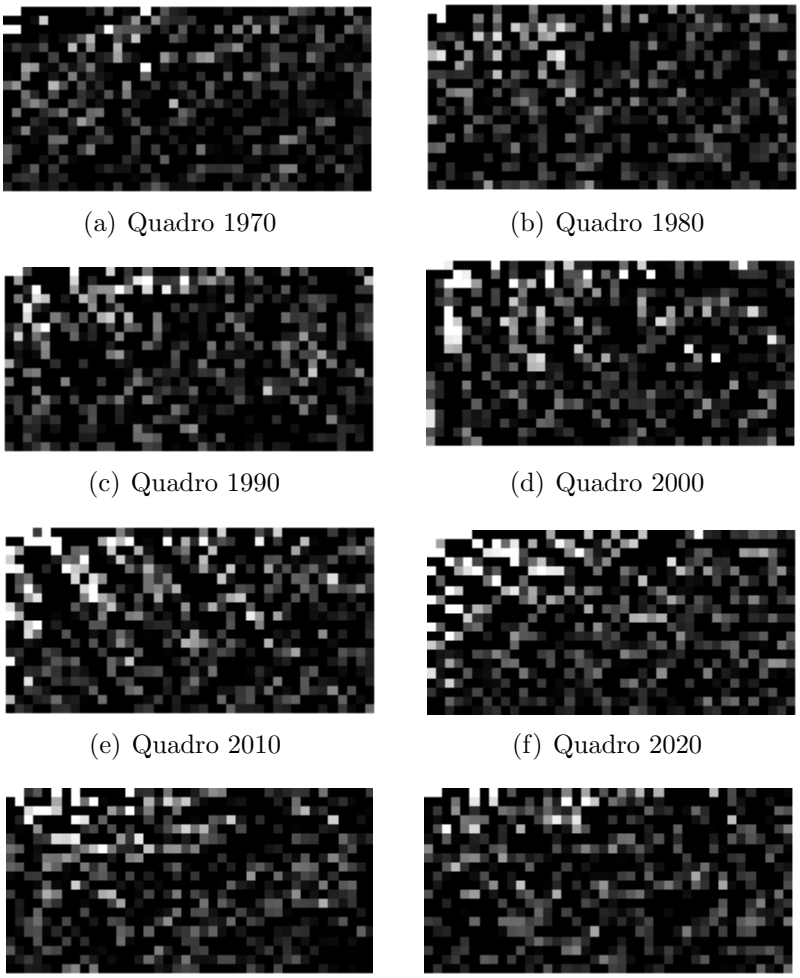
(e) Quadro 2010 (f) Quadro 2020

(g) Quadro 2030 (h) Quadro 2040

Figura 20: DCTs geradas a partir de um trecho que contém um evento

Assim o algoritmo será capaz de buscar variações bruscas na textura do duto e apontá-las como anomalias. A partir dessa etapa o algoritmo passará a utilizar matrizes 20×40 provenientes do canto superior esquerdo da diferença de duas matrizes de DCT referentes a quadros consecutivos.

Para a realização da análise de textura, a matriz foi dividida em quatro setores. Essa divisão foi feita separando as bordas superior e esquerda em quatro partes iguais e considerando os arcos formados pelas 1^{as}, 2^{as}, 3^{as} e 4^{as} divisões de cada borda, como mostrado na Figura 22. Cada setor será usado para definir uma *feature* e estas serão utilizadas para analisar a variação da textura dos quadros. As *features* serão numeradas de 1 a 4 de acordo com o setor utilizado para calculá-las, sendo 1 o setor azul e 4 o setor preto. Cada *feature* é calculada pela soma dos quadrados de todos os pixels do seu setor,



(a) Quadro 1970 (b) Quadro 1980

(c) Quadro 1990 (d) Quadro 2000

(e) Quadro 2010 (f) Quadro 2020

(g) Quadro 2030 (h) Quadro 2040

Figura 21: Janela 20×40 do canto superior esquerdo das DCTs da Figura 20

ou seja:

$$feature_k = \sum_{(i,j) \in setor(k)} M(i,j)^2 \tag{22}$$

onde **M** é a matriz 20×40, k é um número inteiro de 1 a 4 indicando a *feature* e (i,j) é a posição do pixel na matriz **M**.

Após definir as *features* foi necessário verificar como cada uma delas reagia aos eventos no vídeo para poder definir a melhor forma de utilizá-las para detectá-los. Para tal foi escolhido um trecho do duto que contivesse alguns eventos e o algoritmo de detecção do duto foi utilizado para colher as informações necessárias para calcular as *features* de cada quadro. Também foi feita uma análise manual do trecho para verificar a localização dos eventos

Figura 22: Divisão em setores da matriz 20×40 usada na análise de textura

a serem detectados. Essas informações podem ser vistas no gráfico da Figura 23, onde os gráficos das *features* estão em ordem crescente e da mesma cor do setor equivalente da Figura 22. O 5^o gráfico representa a localização dos eventos no vídeo, de acordo com a inspeção manual realizada. Vale ressaltar que, por mais que os eventos estejam indicados em um quadro específico, eles na verdade ocorrem ao longo de vários quadros. Cada um desses pontos indica uma região cuja largura depende da duração do evento.

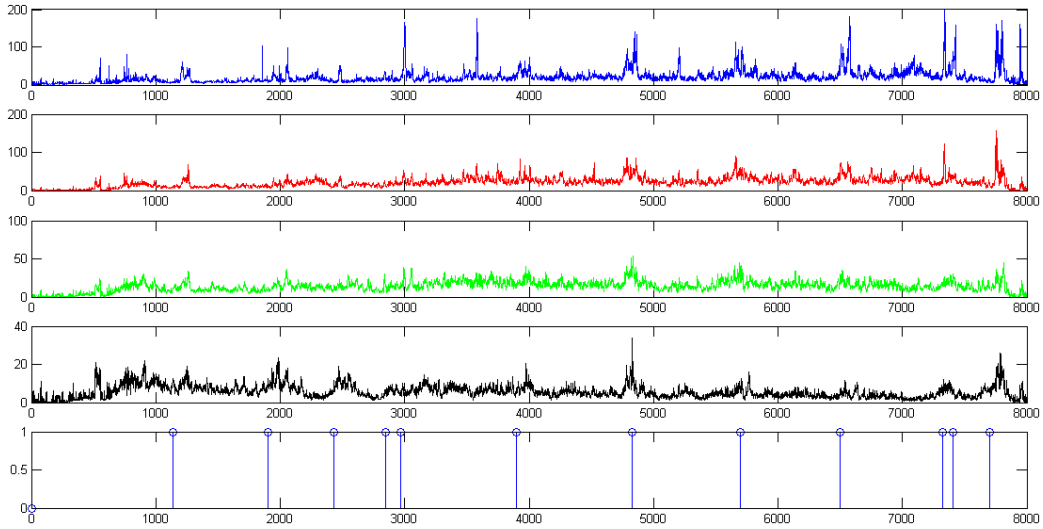


Figura 23: Gráfico das *features* ao longo de um trecho de vídeo contendo alguns eventos

Ao analisar a Figura 23 podemos notar que todas as *features* apresentam picos em algumas das áreas esperadas. Porém a reação da maiorias delas aos eventos é muito inconstante para ser utilizada como detector. As *features*

2, 3 e 4 reagem a muitos dos eventos, porém, como o sinal como um todo é muito ruidoso, seria difícil para o algoritmo detectar essas variações sem elevar muito a quantidade de falsos positivos. Por isso foi definido que apenas a *feature* 1 seria usada pelo detector de eventos.

Observando especificamente o gráfico da *feature* 1, é possível notar que o valor assumido pelo sinal fora dos picos nas diferentes partes do vídeo não é o mesmo; existe um *offset*. Enquanto no início do vídeo o valor da *feature* está próximo de 1 fora dos picos, no meio do vídeo ele se aproxima de 20. Se for utilizado um limiar para definir quais picos são eventos ou não, esse *offset* pode interferir na detecção. Alguns picos apenas ultrapassarão o limiar por já se encontrarem em uma região com *offset*. Caso o limiar seja definido considerando uma região com *offset* então a quantidade de falsos negativos tenderá a aumentar nas regiões com *offsets* menores.

Para contornar o problema do *offset* será utilizada a derivada da *feature* 1. No entanto, essa operação não pode ser realizada diretamente devido ao nível de ruído nesse sinal. Uma diferenciação da *feature* 1 sem um tratamento prévio amplificaria o ruído de tal forma que seria impossível identificar os eventos. Para diminuir o ruído do sinal foi utilizado um filtro de média de 31 pixels de largura seguido de um filtro gaussiano de 21 pixels de largura e σ igual a 5. Após a redução do ruído foi feita a diferenciação do sinal e o resultado foi multiplicado por 10 para facilitar a visualização. Esse vetor será considerado o vetor de variação de textura e pode ser visto no gráfico da Figura 24.

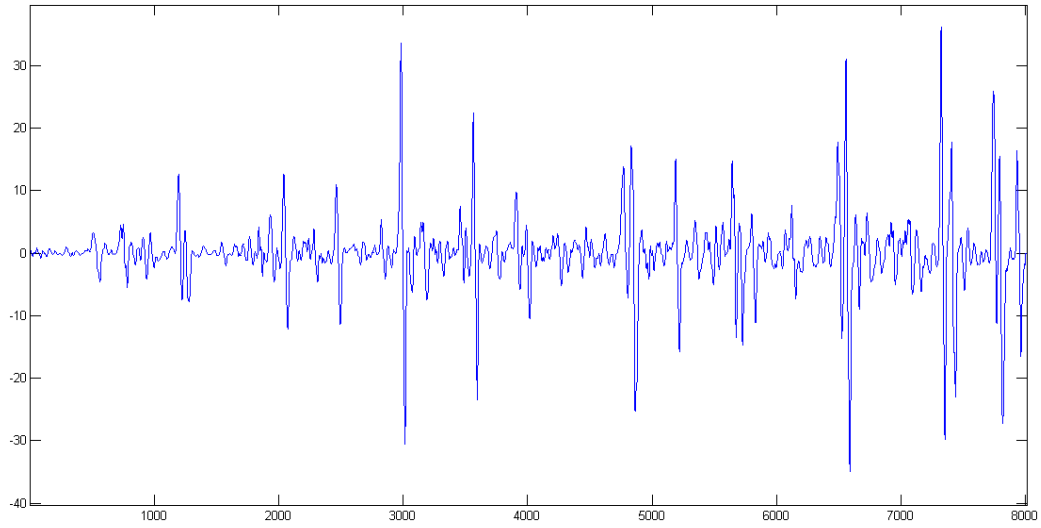


Figura 24: Gráfico da variação de textura no duto ao longo de um trecho do vídeo

Finalmente, é necessário definir um limiar que separará o que será ou não considerado um evento. No caso apresentado como exemplo o limiar escolhido foi 9,5. Ao aplicar esse limiar obtém-se o gráfico da Figura 25, onde o gráfico superior mostra os eventos detectados e o inferior os eventos esperados. No gráfico pode-se ver que o algoritmo foi capaz de detectar quase todos os eventos esperados. Apresentou duas regiões de falso positivo, próximas aos quadros 3500 e 5200, e 1 falso negativo no quadro 2850. Na hora de fazer o ajuste do limiar a ser utilizado é importante lembrar que, devido ao objetivo do programa, a presença de falsos negativos é mais grave que a presença de falsos positivos. A saída do algoritmo será o vetor contendo os *frames* cujo valor de variação de textura ultrapassarem o limiar definido. Essa saída é representada pelo gráfico superior da Figura 25.

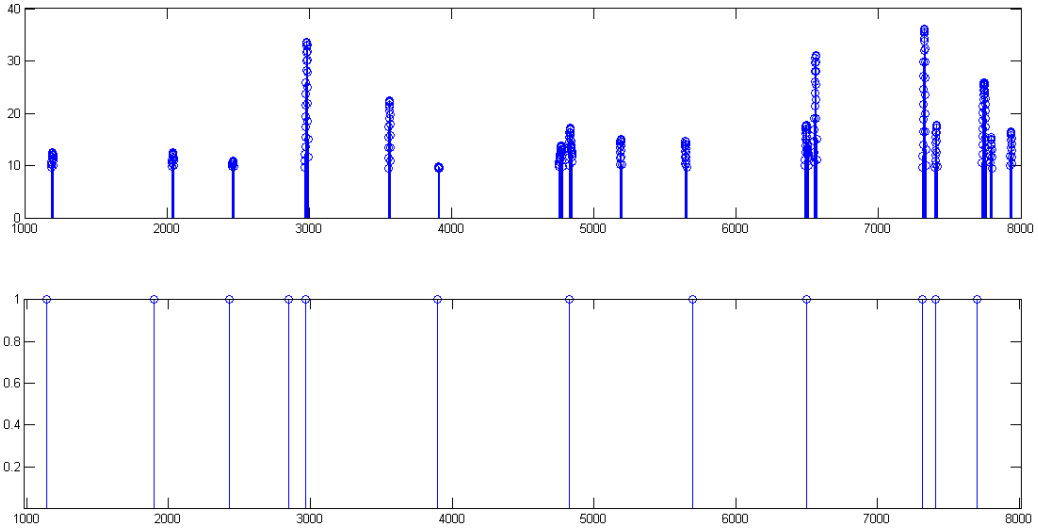


Figura 25: Gráfico das anomalias encontradas pelo programa(superior) e das anomalias encontradas em uma inspeção comum(inferior) ao longo de um trecho do vídeo

6 Resultados

Este trabalho foi capaz de obter resultados satisfatórios com relação ao objetivo a que se propunha. Contudo, ao longo de seu desenvolvimento foram notados possíveis problemas advindos das abordagens escolhidas. Esta seção abordará os resultados obtidos e os problemas encontrados e/ou previstos.

6.1 Detecção do Duto

A detecção do duto foi muito eficiente nos trechos em que foi testada. Estando o vídeo de acordo com as considerações feitas na Seção 3.2.1, o algoritmo é capaz de localizar, com precisão, as bordas do duto ao longo de todo o vídeo.

Além da qualidade da resposta, graças à aplicação local da detecção de bordas, também foi possível aumentar a velocidade do algoritmo. Em testes realizados em um trecho de 4000 quadros, ao utilizar a detecção de bordas na imagem completa, o programa finalizou a análise em 2781s, sendo 1987s dedicados somente à realização da detecção de bordas, 71,5% do tempo total. Utilizando a detecção de bordas localmente, como descrito na Seção 4.2.2, o programa finalizou a análise em 1517s, sendo 685s dedicados à detecção de bordas, 45,2% do tempo total de execução. Graças a essa mudança no algoritmo foi possível diminuir em 45% o tempo total de processamento e em 65,5% o tempo gasto com a detecção de bordas.

Embora eficiente, a abordagem utilizada para melhorar a detecção deixa o algoritmo sujeito a erros muito graves em situações específicas. O maior problema desse algoritmo é que, se, por algum motivo, o algoritmo passar a reconhecer a reta errada, ele dificilmente voltará para a reta correta no futuro. Pode acontecer, por exemplo, do braço do ROV passar a ser detectado como se fosse a reta do duto e, conforme fosse se movendo pela tela, fizesse com que as janelas utilizadas tanto na busca do máximo na matriz de Hough, como na aplicação do detector de bordas, se afastassem dos valores da borda real. A partir desse momento, mesmo que o braço saísse da tela, o algoritmo não seria capaz de encontrar novamente a reta correta. Por isso é muito importante que o vídeo sempre comece em uma situação na qual o duto seja facilmente identificável, uma vez que essa primeira detecção é muito importante para o algoritmo como um todo. Esse algoritmo tem uma chance menor de errar, mas como é extremamente dependente das informações do quadro anterior, não é muito robusto.

6.2 Análise de Textura

Como mostrado na Seção 5 a análise de textura foi capaz de encontrar eficientemente as anomalias presentes no trecho considerado. O trecho utilizado era formado por 8000 quadros, com 12 eventos encontrados em uma inspeção normal. A calibração dos parâmetros do algoritmo foi feita utilizando apenas a primeira metade do trecho, depois a análise foi realizada no trecho completo. Nesse trecho o programa teve 1 falso negativo e 2 falsos positivos, apresentando assim uma taxa de 8,3% de falsos negativos e 16,7% de positivos.

Essas taxas são satisfatórias para uma primeira versão do programa, porém ainda são altas para sua utilização real. Ao analisar os locais em que ocorreram os falsos positivos e negativos, foram descobertos possíveis motivos para suas ocorrências.

No caso do falso negativo, a anomalia em questão se encontrava muito próxima à borda do duto. Como a janela utilizada na análise de textura fica a uma certa distância da borda do duto, uma anomalia que estivesse localizada no canto do duto pode não aparecer na janela avaliada, ou aparecer muito pouco, de forma a não causar uma variação de textura grande o suficiente para ser identificada. Logo, podemos ver que uma distância excessiva entre a janela e as bordas do duto podem dificultar ou impossibilitar a detecção de certos eventos. É necessário um maior estudo a respeito da importância e da frequência de anomalias nas laterais dos dutos, bem como do impacto de uma diminuição excessiva da distância em questão na taxa de falsos-positivos e falsos-negativos.

Quanto aos falsos-positivos, pode ser dito que, na verdade, eles não existiram. A avaliação de quais anomalias devem ou não ser reportadas é muito subjetiva. Isso deve variar com as orientações dadas a cada operador. No caso em questão, ao verificar no vídeo os quadros relativos aos falsos positivos, foram, de fato, encontradas pequenas anomalias que passaram despercebidas na inspeção normal. Pode ser que o operador responsável não considerasse essas anomalias como graves o suficiente para serem reportadas, mas elas existiam. Talvez correspondessem a um estágio inicial de algum problema no duto, mas cabe à empresa responsável pelo duto definir o que ela acha ou não relevante na sua inspeção. Graças a esses casos foi possível ver a importância de uma ferramenta desse tipo. Mesmo considerando um trecho curto do vídeo, pequenos eventos passaram despercebidos pelo avaliador, sendo que esse problema tende a se tornar muito mais grave com o aumento da duração do vídeo.

Além dos casos encontrados, também foram vislumbradas outras situações que poderiam causar falsos positivos. Foi notado que em alguma partes do

vídeo o ROV ajusta seu ângulo de visão. Esses ajustes, feitos por movimentos de rotação, causam no vídeo um movimento muito mais rápido que o visto ao longo do vídeo. É possível que em alguns casos a variação de textura causada por esse movimento anormalmente rápido seja detectada como um evento. Outra possibilidade seria se algo passasse na frente da região do duto que está sendo analisada, como um peixe ou até mesmo o braço do ROV. O aparecimento de um objeto na janela analisada causaria uma grande variação de textura que seria detectada como um evento. O mesmo pode ocorrer caso alguma região do duto apresente uma quantidade anormal de algas.

7 Conclusão

O propósito deste trabalho foi desenvolver um software para auxiliar na inspeção de dutos submersos. Com a aplicação de técnicas de processamento de imagem, o software pode buscar por anomalias no duto e apresentá-las para a avaliação do operador. Dessa forma o operador não necessita analisar o vídeo inteiro, apenas os quadros indicados pelo programa.

Na definição do posicionamento do duto no vídeo foi utilizada uma abordagem para imagens coloridas, aplicando uma variação do detector de bordas de Canny e utilizando a transformada de Hough para definir qual borda pertence ao duto. Para melhorar o desempenho e a velocidade do algoritmo, o detector de bordas foi aplicado apenas nas regiões próximas às bordas encontradas no quadro anterior e a busca dos máximos da transformada de Hough também foi aplicada apenas nas regiões próximas das retas encontradas no quadro anterior. Para a análise de textura foi calculada a DCT de uma região do duto e os coeficientes do canto superior esquerdo da matriz foram utilizados para calcular um índice de variação de textura. Todos os quadros com um índice acima de um determinado limite são reportados como eventos.

O método apresentado se mostrou promissor durante o trabalho realizado e pode tornar a inspeção de dutos mais rápida e segura. No entanto, como dito na Seção 6, ainda existem alguns problemas a serem resolvidos antes que esse método possa ser aplicado na indústria. A existência de falsos-negativos, gerados pela análise de textura ou por uma detecção incorreta do duto, representa um risco para a aplicação imediata do software; enquanto um falso-positivo pode ser facilmente corrigido pelo operador, a correção de um falso-negativo exigiria uma quase completa reavaliação do vídeo.

7.1 Trabalhos Futuros

Trabalhos futuros deveriam explorar soluções para os problemas citados na Seção 6. Sugerimos os seguintes tópicos:

- Desenvolvimento de um método para identificar erros na detecção de bordas;
- Método para correção de erros na detecção de bordas;
- Redução dos falsos-negativos na análise de textura.

Referências

[1] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6), 1986.

[2] Rafael C. Gonzalez and Richard E. Woods. *Processamento Digital de Imagens*. Pearson Prentice Hall, 3a edition, 2010.

[3] Paul Hough. Methods and Means for Recognizing Complex Patterns. *U.S. Patent 3,069,654*, 1962.

[4] Syed Ali Khayam. The discrete cosine transform (dct): theory and application. *Michigan State University*, 2003.

[5] Andreas Koschan and Mongi Abidi. Detection and classification of edges in color images. *Signal Processing Magazine, IEEE*, (January 2005):64–73, 2005.

[6] Andrzej Materka and Michal Strzelecki. Texture analysis methods-a review. *Technical University of Lodz, Institute of Electronics*, 1998.

[7] Steven A. Shafer. Using color to separate reflection components. *Computer Science Department, University of Rochester*, 1984.

[8] Joost van de Weijer, Theo Gevers, and Jan-Mark Geusebroek. Edge and corner detection by photometric quasi-invariants. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(4):625–30, April 2005.

[9] Joost van de Weijer, Theo Gevers, and Arnold W. M. Smeulders. Robust photometric invariant features from the color tensor. *IEEE Transactions on Image Processing*, 15(1):118–27, January 2006.

[10] Joost Van De Weijer. Color edge detection by photometric quasi-invariants. *Proceedings Ninth IEEE International Conference on Computer Vision*, (Iccv):1520–1525, vol.2, 2003.