Universidade Federal do Rio de Janeiro
Escola Politécnica
Departamento de Engenharia Eletrônica e de Computação

José Felipe Torres Lima

DESENVOLVIMENTO DO TRATAMENTO DE DADOS NO PROGRAMA REAL TIME SCAN

UFRJ

José Felipe Torres Lima

DESENVOLVIMENTO DO TRATAMENTO DE DADOS NO PROGRAMA REAL TIME SCAN

Dissertação de Graduação apresentada ao Departamento de Engenharia Eletrônica e de Computação, Escola Politécnica, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Graduando em Engenharia Eletrônica e de Computação.

Orientador: Antonio Petraglia

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO

Escola Politécnica – Departamento de Eletrônica e de Computação
Centro de Tecnologia, bloco H, sala H-217, Cidade Universitária.
Rio de Janeiro – RJ CEP 21949-900

AGRADECIMENTO

Agradeço principalmente aos meus pais, por me dar a vida, educação e consciência. Aos meus amigos, irmãos e irmãs, por abrir meus olhos a um mundo único de experiências. Agradeço também aos meus professores, pelo conhecimento compartilhado, e oportunidades geradas. À Minds at Work e ao CPTI agradeço pelo trabalho que gerou esse Projeto Final e pela camaradagem da equipe fantástica. Por fim, agradeço a esse universo maravilhoso, cheio de belezas, encontros inesperados e felicidades.

RESUMO

LIMA, José Felipe Torres. **Desenvolvimento do tratamento de dados no programa Real Time Scan.** Rio de Janeiro, 2012. Dissertação (Graduação) – Departamento de Engenharia Eletrônica e de Computação, Escola Politécnica, da Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2012.

O RTScan é um software de análise de dados resultantes de inspeções de dutos, criado pelo CPTI em parceria com a empresa Minds at Work. Em sua primeira versão, o RTScan utiliza uma estrutura de processamento sequencial, dificultando a análise dos dados gerados pelo PIG, pois sua velocidade é muito baixa. Com o avanço da tecnologia que permitiu que os processadores tivessem diversos núcleos e a facilidade em programar estruturas paralelizáveis promovida pela utilização da biblioteca Qt4, tomou-se a decisão de se criar uma nova versão do RTScan onde o processamento da informação é realizado em paralelo. Um grupo de engenheiros e programadores ficou incumbido de criar esta nova versão multi-thread, e desenvolver as implementações necessárias para otimizar a análise gráfica dos dados.

Este trabalho relata o desenvolvimento das classes do software responsáveis pelo tratamento dos dados brutos, gerando gráficos para a análise do CPTI. Minha participação no projeto em parceria com a Minds at Work foi o de criar tais estruturas que processassem os dados de forma que retratassem com fidelidade o duto, facilitando a detecção de eventos durante as análises. Configuradores, Condensadores e pequenos blocos de processamento, denominados Plug-ins foram desenvolvidos para realizar tal tarefa.

Como resultado das implementações-alvo deste trabalho, este relatório apresenta em sua conclusão uma comparação da análise gráfica possivel no momento anterior às modificações e aquela com a utilização do software atual.

Palavras-Chave:
Aceleração por paralelismo; Inspeção de Dutos; PIGs Instrumentados;

ABSTRACT

LIMA, José Felipe Torres. **Desenvolvimento do tratamento de dados no programa Real Time Scan**. Rio de Janeiro, 2012. Dissertação (Graduação) – Departamento de Engenharia Eletrônica e de Computação, Escola Politécnica, da Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2012.

The RTScan is a data analysis software for inspections of pipelines, created by CPTI in partnership with Minds at Work. In its original version, the RTScan uses a sequential processing structure, which increases the processing time, slowing the analysis of data generated by the PIG. With the advancement of technology that allowed processors to have multiple cores and with the easiness in programming parallelizable structures promoted by the use of Qt4 library, a new version of RTScan was created, in which parallel data processing was carried out. A group of engineers and programmers was assigned with the task of creating this new multi-thread version, and of developing the necessary implementations to optimize graphical data analysis.

This work reports the development of the software's classes responsible for processing the raw data, generating graphs for CPTI analysis. My participation in the project in partnership with Minds at Work was to create this structures for data processing so that it could faithfully portray the pipeline, facilitating the detection of events during the analyzes. Configurators, Condensers and small processing blocks, called Plug-ins have been developed to perform this task.

As a result of the target implementations, the conclusion of this report presents a graphical comparison of the analysis time prior to the modifications and the one using the current software.

Keywords:
Acceleration by parallelism; Pipeline Inspection; Instrumented PIGs;

SIGLAS

PUC – Pontifícia Universidade Católica do Rio de Janeiro

PIG – Pipeline Inspection Gauge (Ferramenta para Inspeção de Dutos)

US – Ultrassom

RTScan – Real Time Scan (Inspeção em Tempo Real)

CENPES – Centro de Pesquisas da Petrobrás

A/D – Analógico – Digital

ASME – American Society of Mechanical Engineers (Sociedade Americana de Engenheiros Mecânicos)

ASME B31G – Manual para determinar a força restante de um duto corroído

RSTRENG – Tecnologia para determinar a força restante de um duto externamente corroído

CSV – Comma Separeted Values (Valores separados por vírgula)

UFRJ – Universidade Federal do Rio de Janeiro

HSV – Hue, Saturation and Value (Matiz, Saturação e Valor)

UI – User Interface (Interface para o Usuário)

XML – eXtensible Markup Language (Linguagem de Marcação Extensível)

CPTI – Centro de Pesquisas em Tecnologia de Inspeção da PUC-Rio

TS – Timestamp

SUMÁRIO

1 Introdução

Com mais de 15 mil quilômetros de extensão, a rede nacional de dutos do setor petrolífero pode ser comparada ao sistema circulatório humano. Só que, em lugar de veias e artérias, ela é composta de tubulações metálicas de vários diâmetros, por onde circulam óleo, derivados e gás natural. Com o passar dos anos de operação, estes dutos apresentam diversos problemas como: corrosão, depósitos de substâncias dissolvidas no petróleo e soldas defeituosas que podem causar dificuldades como: retenção de fluxo e, em situação extrema, rompimento com vazamentos para o meio ambiente. Manter essa malha funcionando requer complexos sistemas de monitoramento e reparo, onde os ensaios não destrutivos vêm sendo desenvolvidos e aprimorados em centros de pesquisas, como uma ferramenta fundamental nas inspeções periódicas, para assegurar a integridade da parede dos dutos.

O CPTI, em parceria com o CENPES, vem desenvolvendo um sistema de Ensaio Não Destrutivo composto por um dispositivo autônomo de coleta de dados, chamado PIG, e por um software de análise de dados, o RTScan.

Os PIGs são robôs, equipados com sensores avançados usados para limpar tubulações e identificar problemas estruturais, aumentando a eficiência dos ensaios e reduzindo os seus custos. Um PIG não possui propulsão própria sendo carregado pelo fluxo do fluído. Existem diversos tipos de PIGs variando de acordo com: as diferentes dimensões dos dutos, o tipo de fluido transportado e o tipo de instrumentação do mesmo. Maiores detalhes serão descritos no Capítulo 2.

Uma inspeção também denominada corrida, começa com o PIG sendo introduzido numa extremidade do duto através de um sistema de válvulas e sendo recuperado na outra. As amostras de dados colhidas pelo PIG, em um intervalo de tempo pré-estabelecido da ordem de milisegundos, são gravadas em um arquivo no seu dispositivo eletrônico. Tais amostras relacionam-se à intensidade dos sinais obtidos pelos sensores e à posição da medida. A dimensão do arquivo pode atingir gigabytes, variando de acordo com o comprimento do duto. Estes dados são processados pelo programa RTScan.

O RTScan é um programa elaborado em linguagem C++ que utiliza a biblioteca Qt. Na versão antiga que ainda está em operação, o programa apresenta os dados em tela para o operador de forma lenta, dificultando consideravelmente a análise dos mesmos e fazendo com que a inspeção de uma corrida possa ter a duração de alguns dias. Para isso uma nova versão do software está sendo desenvolvida onde toda a estrutura será modificada. Maiores informações serão detalhadas no Capítulo 3.

Este projeto tem como objetivo principal o tratamento dos dados obtidos no ensaio utilizando a nova versão do software RTScan, de modo que os dados possam ser visualizados graficamente segundo a intensidade de sinal e a posição física dentro do duto. Consequentemente, qualquer alteração física no interior do duto ganhará destaque através da visualização gráfica e propiciará ao operador rapidez e facilidade de análise dos resultados. Esses tratamentos serão melhor descritos no Capítulo 4.

Finalmente, no Capítulo 5, serão apresentadas as conclusões.

Capítulo 1 PIG

3.1 Introdução

Os PIGs instrumentados são robôs de tamanhos variados amplamente utilizados nas inspeções de dutos. Seus trabalhos baseiam-se em diferentes tipos de técnicas de medição, seja para detecção de perda de espessura por corrosão interna e/ou externa e amassamentos, obstruções/acúmulos ou vazamentos no duto.

Os PIGs são constituídos por módulos com vasos de pressão sustentados por copos de poliuretano, dentro dos quais está instalada uma eletrônica embarcada de aquisição de dados e uma bateria, contando ainda com hodômetros para registro da distância e posição. Um exemplo ilustrativo de PIG instrumentado é mostrado na Figura 2-1.

Estes equipamentos são deslocados dentro do duto a partir da própria energia de bombeio do fluido interno. A diferença de pressão à jusante e à montante dos copos de tração do PIG, criada no transporte do fluido, impulsiona o mesmo pelo duto.

Existem diversos tipos de PIGs variando de acordo com as diferentes dimensões dos dutos, o tipo de fluido transportado e o tipo de instrumentação. Os tipos de instrumentação incluem sensores mecânicos, ultrassônicos, magnéticos e de radiação infravermelha. Um sensor consegue analisar a parede do duto apenas num ponto próximo a ele e, por isso, um PIG é equipado com um grande número de sensores geralmente dispostos em conjuntos circulares denominados coroas. Uma coroa é um grupo de sensores posicionados um ao lado do outro com distâncias fixas, apresentando um deslocamento angular entre elas para aumentar a resolução espacial da análise.

Cada inspeção ou corrida começa com o PIG sendo introduzido numa extremidade do duto através de um sistema de válvulas e sendo recuperado na outra com os dados coletados armazenados à sua eletrônica. O arquivo gerado pode ter o tamanho de gigabytes e os dados de cada corrida são analisados com a finalidade de relatar a localização de eventuais defeitos na parede do duto, bem como suas posições horárias e longitudinais. Assim, baseando-se no relatório pós-inspeção do PIG instrumentado e de acordo com as normas e com os critérios de avaliação de dutos, como ASME B31G ou RSTRENG, os engenheiros responsáveis pelo duto decidem quais ações preventivas e/ou corretivas devem ser realizadas.

Figura 1-1 Exemplo de PIG instrumentado.

Como todo equipamento de precisão, o PIG precisa ser aferido para medição correta de uma grandeza física. Os dados coletados do PIG precisam ser processados, considerando as configurações dos seus sensores, para produzirem representações fieis do duto e de seus defeitos. Há duas principais caracteristicas do PIG que impedem que os dados coletados na

corrida sejam referências diretas à profundidade do defeito e à sua posição no duto. A primeira é a disposição dos sensores em formato de coroas e a segunda é a necessidade dos sensores serem calibrados.

3.2 Disposição em Coroas

Para atender uma resolução circunferencial (radial) milimétrica, o PIG é dotado de mais de uma coroa instrumentada de sensores ao longo do seu corpo. A utilização de diversas coroas ocorre porque mesmo que quiséssemos colocar milhares de sensores um ao lado do outro, isso só geraria uma medição perfeita caso os sensores fossem milimétricos. Assim sendo, essas coroas instrumentadas são angularmente defasadas umas das outras, como pode se notar na Figura 2-2, para distribuição uniforme de sensores ao longo do perímetro da superfície interna do duto, para que o operador possa visualizar com maior definição a circunferência do duto. Porém, pelo fato delas serem defasadas também em relação ao comprimento do duto, os dados obtidos não representam exatamente aquela posição adquirida pelo hodômetro.

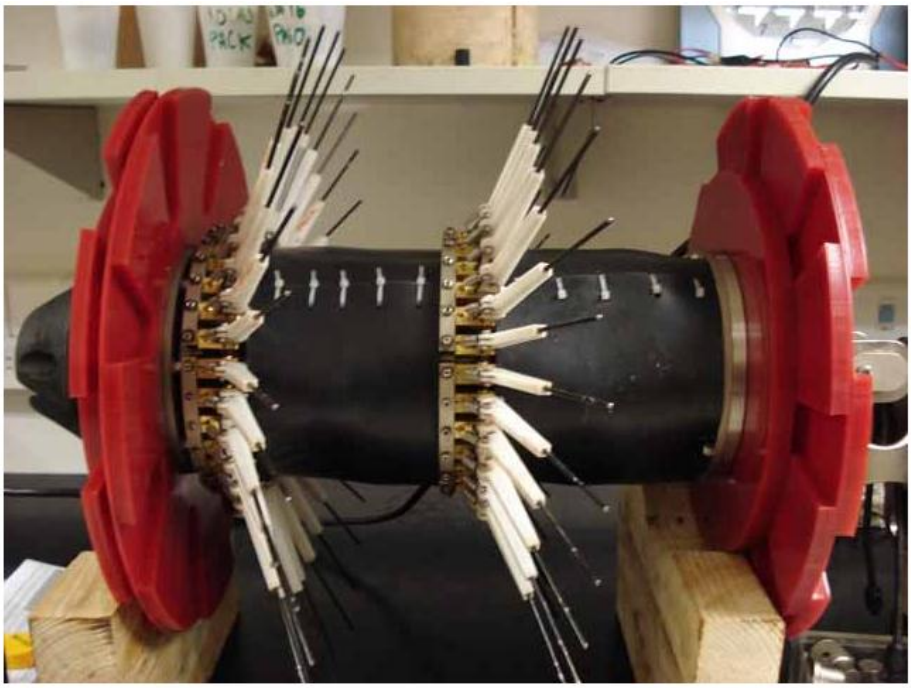

Figura 1-2 Posicionamento de duas "coroas" de sensores "palito".

Além disso, outro importante problema a ser ressaltado é que a placa conversora A/D geralmente fica no centro do PIG, distante de seus sensores. A placa é conectada atraves de cabos, que, para redução de ruídos, são o mais curtos possiveis, de forma que todos os canais em uma coroa estão ligados em sequência na placa A/D. Essa configuração dos sensores resulta em dados consecutivos que não podem ser relacionados a ângulos sequenciais.

Esses dois empecilhos para visualização perfeita do duto, gerados pela estrutura mecânica do PIG, serão futuramente corrigidos pelo software RTScan no momento do tratamento dos dados

3.3 Tipos de PIGs

Descreveremos a seguir os dois principais tipos de PIGs utilizados no estudo e suas calibrações.

3.3.1 PIG Ultrassônico

Ilustrado na Figura 2-3, o PIG Ultrassônico (PIG US) usa sensores acústicos que medem a diferença do tempo de propagação dos ecos refletidos nas superfícies das paredes interna e externa do duto. Normalmente essas ferramentas possuem três ou mais coroas, onde os sensores ultrassônicos ficam distribuídos ao longo do corpo do PIG. O PIG US mede o perfil de espessura do duto ao longo da inspeção. A eletrônica embarcada do PIG US é geralmente mais complexa que outros PIGs instrumentados convencionais, pois precisa provocar e detectar o sinal ultrassônico numa taxa de aquisição elevada. O PIG US detecta e dimensiona perdas de espessura provocadas por corrosão interna ou externa, traçando o perfil de espessura remanescente do duto. Outros tipos de defeitos em dutos, como trincas, são também detectadas pela técnica de ultrassom.

Figura 1-3 PIG Ultrassom durante sua calibração.

A tecnologia utilizada pelo PIG US convencional produz resultados excelentes dependendo da sua aplicação. No entanto o sucesso da inspeção depende diretamente do fluido do duto. A técnica de medição por ultrassom necessita de um meio homogêneo para a propagação das ondas acústicas, o que nem sempre é comum de se encontrar em dutos de petróleo. O escoamento multifásico de fluidos é predominante nos campos de produção

offshore de petróleo, e existem situações onde é possível encontrar água, óleo e gás num único produto. Uma solução típica é a introdução de um "colchão de diesel" para servir de fluido acoplante durante a inspeção, mas este tipo de operação só se torna possível com o processo de escoamento e tratamento do fluido.

3.4.1.1 Calibração do PIG Ultrassônico

A calibração do PIG Ultrassônico é bem simples. Nela verifica-se a velocidade que o som percorre em cada uma das camadas das superficies que se deseja medir. Por isso a calibração geralmente é feita colocando-se o PIG US dentro de uma câmara como a da Figura 2-2, feita do mesmo metal do duto, preenchida com o líquido que irá impulsionar o PIG. Nesta estrutura conhecida são determinados os tempos de retorno dos pulsos que os sensores emitem.

3.4.2 PIG Palito

O PIG Palito usa apalpadores instrumentados. Estas estruturas mecânicas são basicamente compostas de um palito de contato fixo dentro de uma haste articulada, um eixo de rotação desta haste, que está fixa à base do sensor, e o transdutor de efeito Hall dentro deste eixo. Imãs, que fornecem um campo magnético constante ao conjunto eixo e transdutor, estão fixos dentro da haste articulada do sensor palito. A simplicidade desta técnica de medição e o baixo custo de produção deste PIG em relação aos demais são vantagens que tornam essa nova tecnologia uma alternativa em inspeções de detecção e dimensionamento de corrosão interna em dutos. Um exemplo ilustrativo de PIG Palito é mostrado na Figura 2-4.

Figura 1-4 Exemplo de PIG Palito

3.4.2.1 Calibração do PIG Palito.

A calibração do PIG Palito é um processo mecânico e manual operado por técnicos, que utilizam o mesmo gabarito de calibração individualmente em cada sensor. Este processo visa aferir todos os sensores, para que posteriormente se possa utilizar o equipamento na

inspeção do duto. Isto acontece porque o processo de montagem, o ajuste mecânico e inevitáveis diferenças dimensionais de fabricação na usinagem de peças mecânicas podem introduzir imprecisões na medição da micro-geometria da superfície do duto.

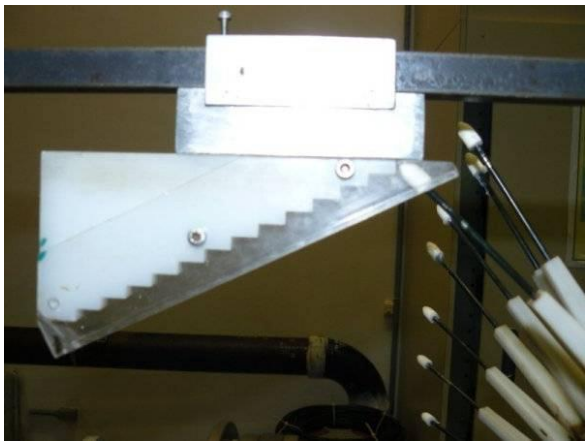

Figura 1-5 Sensor palito sendo aferido no primeiro degrau do gabarito de calibração.

Os componentes utilizados na calibração são um gabarito de calibração, um exemplo pode ser visto na figura 2-5, e um trilho guia de deslizamento do gabarito, um exemplo pode ser visto na figura 2-6. O primeiro componente, o gabarito de calibração, é uma peça mecânica utilizada como referência dimensional da faixa de excursão do sensor, que possui degraus usinados com alturas definidas a partir do raio do centro do PIG. A segunda peça, o trilho guia, é montada também em função do raio do corpo do PIG e está fixo ao suporte do vaso e/ou corpo instrumentado para centralizar e realizar o escorregamento do gabarito de calibração.

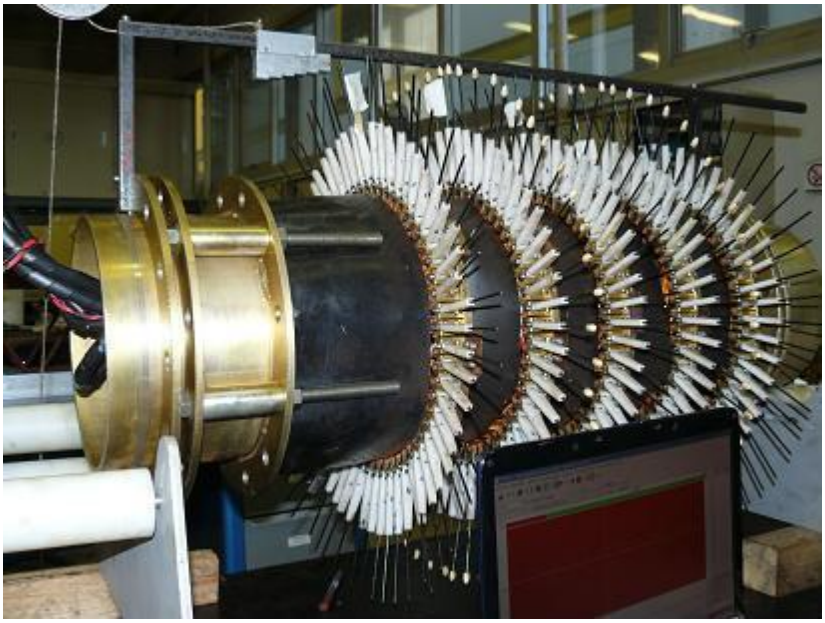

Figura 1-6 Calibração de um PIG Palito

Capítulo 2 RTScan – Real Time Scan

4.1 Introdução

O software tem uma versão anterior. Por ser uma aplicação single-thread, o processamento nesta versão é lento, aumentando o tempo de uma análise. Nesta nova versão estamos remodelando completamente a estrutura do software, de forma que o processamento dos dados e comunicação dos componentes seja multi-thread. O programa é desenvolvido por uma equipe de programadores que trabalha para o CPTI. Nessa equipe cada um é responsável por uma área de desenvolvimento. Este projeto tem como tema central , o de desenvolver os Plug-ins, componentes que processam os dados brutos vindos do PIG e os apresentam em tela para o operador. Primeiramente uma descrição geral do funcionamento do programa será apresentada e, mais a frente, a descrição de cada Plug-in e de suas ferramentas, trabalho este que foi desenvolvido pelo autor deste trabalho.

4.2 Estado Original da nova versão do Software RTScan

No programa encontramos diversos componentes chave, a RTScanMainWindow, o Manager, as ConfigKeys, o Resolvedor, os Plug-ins, a BlkApi e os Configuradores.

A RTScanMainWindow é a responsável por tratar da interface visual do programa. Fora a linha de comando, é através dela que o usuário tem como interagir com o programa. Depois de decidida qual corrida será analisada a RTScanMainWindow informa ao Manager, um membro que funciona em paralelo com as interações do analista.

O Manager é o gerenciador que faz as ligações necessárias entre o processamento e a visualização. O Manager possui uma instância de ConfigKeys, o componente de comunicação, que funciona basicamente como uma tabela associativa em que as chaves são strings e os valores podem ser de qualquer tipo. Cada uma dessas chaves contém uma configuração relacionada à corrida e qualquer alteração no conjunto das chaves ou no seu valor dispara notificações para os componentes interessados. O ciclo de vida do Manager está associado a uma corrida aberta. Quando a corrida é aberta, o Manager busca pelo arquivo de ConfigKeys associado à corrida e, se este existir, restaura todas as chaves geradas na ultima abertura da corrida. Na destruição do Manager, ele salva o ConfigKeys para um arquivo de extensão .ck com nome idêntico ao do arquivo da corrida.

Ao iniciar, o Manager também inicia a BlkApi, um tratador do arquivo .blk. A princípio esse componente informa o tipo de corrida (Palito ou US) e a estrutura dos dados no arquivo ao Resolvedor, um outro componente do programa que identifica quais Plug-ins são necessários para os dados brutos serem processados em dados para a tela, montando assim uma cadeia de Plug-ins associada a uma corrida aberta. Um exemplo de cadeia do estado original pode ser visto na Figura 3-1. Para cada Plug-in da cadeia, o Manager acessa as chaves do ConfigKeys das quais ele depende. Ele também passa a referência para a transação corrente do ConfigKeys de forma que o Plug-in possa verificar as modificações nas chaves. Como na primeira abertura da corrida as chaves ainda não existem, neste momento, o Plug-in tem a oportunidade de armazenar um valor default para as chaves das quais ele necessita.

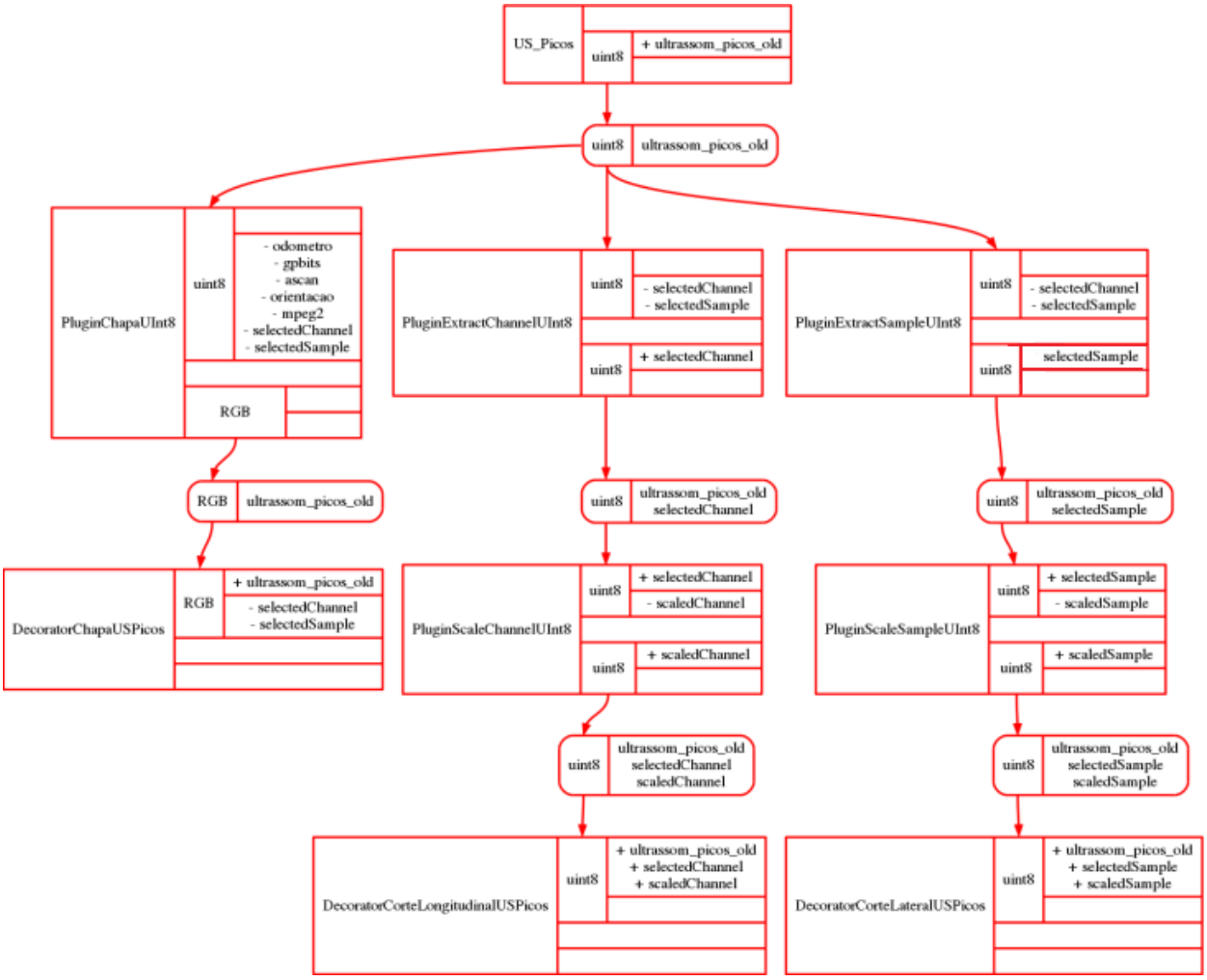

Figura 2-1 Árvore de Plug-ins

Tendo a lista de todas as chaves em mãos, o Manager verifica quais delas necessitam de Configuradores, componentes que têm como função modificar as chaves, e os instancia, passando a referência para a transação do ConfigKeys corrente. Durante a execução, os Configuradores alteram as chaves e podem concluir a transação ou não. Da forma como as referências para transações funcionam, quando uma transação é concluída, ela é guardada numa pilha para que se torne passível de um "desfazer" – "refazer". Como diversos componentes tinham referência à área de memória dessa transação concluída, as referêcias passam a apontar para uma nova transação. Todas as alterações efetuadas na transação são visíveis a todos os componentes que possuem uma referência a ela.

Cada alteração efetuada na transação é conectada ao método update do Manager, um método que localiza qual área em tela precisa ser redesenhada. Para isso o Manager gera um pedido à BlkApi, componente que trata do arquivo blk. Nesse pedido ele informa que espectro de amostras que ele deseja e o nivel de discretização encontrado em tela. Os dados necessários para preenchimento da tela são entregues ao Condensador, componente que irá escolher a melhor amostra para determinado pixel, sendo que, neste momento, ele escolhe a primeira. Depois de condensados, os dados se encontram num buffer de amostras e o Manager indica quais Plug-ins necessitam de atualização e em que área do buffer devem trabalhar. Quando esses Plug-ins são invalidados, eles precisam ser executados novamente. Com o

encadeamento deste processo, quando um Plug-in sofre uma invalidação, todos aqueles acima dele na cadeia também são invalidados, de forma que a tela é atualizada a cada mudança nas configurações.

Originalmente o programa apresentava a informação, conforme mostrado na Figura 3-2. O Software tinha seis Plug-ins e com eles os dados eram processados e exibidos em três gráficos diferentes, a chapa, o corte lateral e o corte longitudinal, essas estruturas ainda são utilizadas, porém com melhorias. A chapa é um gráfico onde o dado é representado por tons de cor. Por exemplo, ao utilizarmos tons de cinza, a cor preta correspondente ao valor máximo e a branca ao valor mínimo. Os cortes são gráficos de secção. Na chapa há um marcador de referência, comumente conhecido como "mosca", para selecionar um canal e uma amostra, o corte lateral mostra todos os canais da amostra selecionada e o corte longitudinal mostra o valor do canal selecionado de cada uma das amostras em tela.

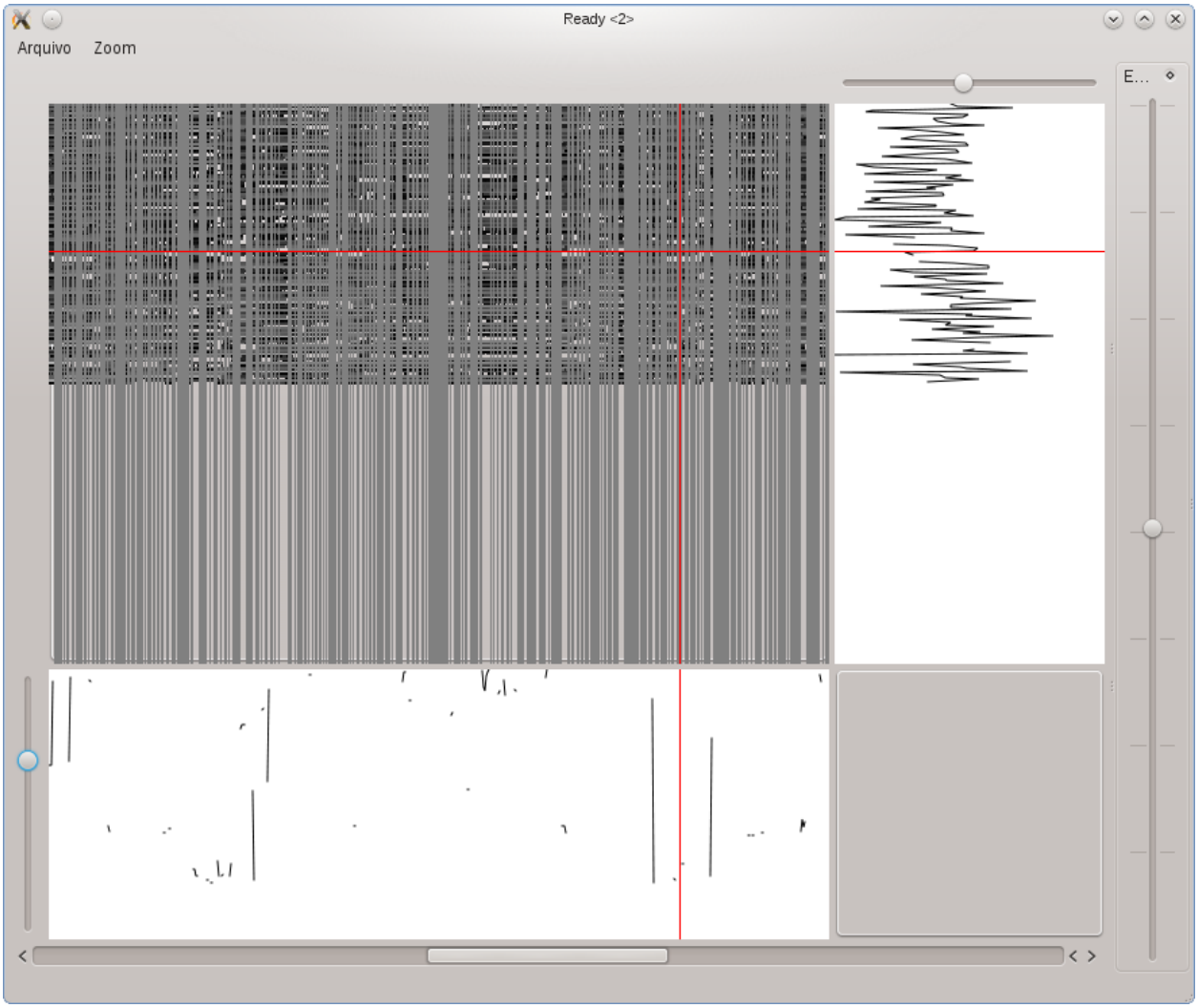

Figura 2-2 Estado original do software

Tínhamos somente seis chaves no ConfigKeys: dicretização, posiçãoCentral, canalAtual, amostraAtual, escalaLongitudinal e escalaLateral. Existiam também três Configuradores de escala que modificam as chaves: escalaLongitudinal, escalaLateral e escalaCor.

O usuário do programa só tinha quatro possíveis interações com o programa:

- Modificar a posição da corrida, avançando ou retrocedendo no duto.
- Aumentar ou diminuir o nível de discretização da corrida (Zoom-In, Zoom-Out).
- Mudar o Canal e Amostra atual, possibilitando ver secções diferentes do duto.
- Modificar o ganho nos cortes e na chapa dando mais ênfase nos defeitos.

Cada uma dessas interações modifica uma chave e cada uma dessas desencadeia um processo de invalidação de Plug-ins. Por exemplo, sempre que canalAtual é modificada o Plug-in CorteLogitudinal precisa se reprocessar, analogamente ao caso da amostraAtual e o Plug-in CorteLateral.

O programa ainda se encontra num estado que não pode ser utilizado por um analista. Os dados em tela ainda necessitam de um tratamento específico antes de representarem fielmente o duto. E para isso este projeto deseja implementar todos os Plug-ins necessários o RTScan possa ser utilizado pelos analistas do CPTI.

4.3 Estrutura de um arquivo BLK

Os dados coletados de uma corrida são armazenados em blocos de tamanho fixo para aquela corrida (tipicamente de 128 KB a 2 MB). Cada bloco tem, necessariamente, um cabeçalho seguido de uma ou mais amostras. Pode haver um pequeno espaço não aproveitado no final de cada bloco.

O cabeçalho de bloco contém todos os dados necessários para a:

- identificação da corrida do bloco
- decodificação dos dados armazenados
- procura de uma amostra por número de amostra, tempo ou posição (por posição é opcional)

As amostras de um bloco são organizadas por ordem crescente de tempo. Cada amostra registra dados adquiridos idealmente em um mesmo instante, bem como o instante em que os dados foram coletados.

4.3.1 Tipo de dado

Conceito determinado pelo software de tratamento, ignorado pelo PIG e teoricamente independente do formato dos dados coletados (conceito introduzido adiante). Para interpretar os tipos de dados em uma corrida é necessário combinar os dados coletados aos dados de configuração do PIG. São exemplos de tipos de dados os associados a odômetros digitais, odômetros analógicos, bobinas magnéticas, sensor Hall de corrosão, sensor Hall de detecção de soldas, palhetas de geometria, juntas, acelerômetros, giroscópios, sinais de controle ambiental da eletrônica, temperatura e pressão do fluido, sensor de orientação horária.

4.3.2 Formato de dado

Especificação válida para todas as corridas, identificada por um número de código (int16), que determina como um dado é armazenado. Os tipos de dados são associados aos formatos de dados pelos programas de tratamento e pelos dados de configuração do PIG. Além do byte de identificação, um formato de dado tem um int16 N denominado Complemento, que completa sua especificação. Este complemento guarda informações adicionais do formato de dado, que não estão previstas na especificação do formato.

Alguns formatos suportam uma representação especial que é colocada no espaço reservado existente no cabeçalho do bloco. Esta representação especial não deve ser considerada como uma amostra do arquivo. Ela existe independentemente da presença de amostras e possui informações adicionais relacionadas com o formato, como no caso dos formatos compactados por diferença. Nestes a amostra representa a diferença do valor atual para a amostra anterior e não o valor absoluto. Torna-se necessário introduzir uma representação especial com um "offset" inicial para que o valor absoluto possa ser calculado para todas as amostras.

Atualmente, os seguintes Formatos de Dados estão definidos. Futuramente, poderão ser definidos formatos adicionais, sem que as especificações para os formatos abaixo sejam alteradas.

4.3.2.1 Formato 0

Complemento: N entre 1 e 8
Valor: uint32[N]
Tipos associados: odômetros digitais
Representações especiais: uint32[N], representando o número de transições no início do bloco e uint32[N], representando o número de transições no final do bloco.
Representação no bloco: 1 byte onde 1 no bit i representa um incremento em uint32[i]

4.3.2.2 Formato 1

Complemento: irrelevante (0)
Valor: cstring
Tipos associados: mensagens
Representação no bloco: cstring (terminada em zero)

4.3.2.3 Formato 2

Complemento: N > 0
Valor: int8[N]
Tipos associados: sinais representáveis em 8 bits (signed ou unsigned char)
Representação no bloco: int8[N]

4.3.2.4 Formato 3

Complemento: N > 0
Valor: int8[N]
Tipos associados: sinais representáveis em 8 bits (signed ou unsigned char)
Representação especial: int8[N], representando o valor absoluto do sinal no início do bloco. Obs.: Caso não tenha sido feita nenhuma aquisição no momento que o cabeçalho é montado (primeiro bloco da corrida) este valor pode ter um valor arbitrário a critério do software como, por exemplo, o zero ou o meio da escala.

Representação no bloco: Seqüência de nibbles representando de int8[0] a int8[N-1], com número variável de nibbles, implementando um esquema de compactação por diferenças em relação ao item int8[i] da amostra anterior. Para a primeira amostra a diferença é sobre o valor contido na representação especial.

4.3.2.5 Formato 4

Complemento: N > 0

Valor: int16[N]
Tipos associados: sinais representáveis em 16 bits (signed ou unsigned char)
Representação no bloco: int16[N]. Byte menos significativo primeiro.

4.3.2.6 Formato 5

Complemento: N > 0
Valor: uint16[N]
Tipos associados: sinais representáveis em 16 bits (signed ou unsigned char)
Representação especial: uint16[N], representando o valor do sinal no início do bloco. Byte menos significativo primeiro. Obs.: Caso não tenha sido feita nenhuma aquisição no momento que o cabeçalho é montado (primeiro bloco da corrida) este valor pode ter um valor arbitrário a critério do software como, por exemplo, o zero ou o meio da escala.

Representação no bloco: Seqüência de nibbles representando de int16[0] a int16[N-1], com número variável de nibbles, implementando um esquema de compactação por diferenças em relação ao item uint16[i] da amostra anterior. Para a primeira amostra a diferença é sobre o valor contido na representação especial.

4.3.2.7 Formato 6

Complemento: N > 0, N é múltiplo de 4
Valor: int10[N]
Tipos associados: sinais representáveis em 10 bits (signed ou unsigned char)
Representação no bloco: Seqüência de 5 bytes (B0, B1, B2, B3, B4 E B5) representando 4 dados (S0, S1, S2 e S3 - ver figura abaixo). O byte B0 armazena do bit 9 ao bit 2 do dado S0; O byte B1 armazena do bit 1 ao bit 0 do dado S0 e do bit 9 ao bit 4 do dado S1, e assim por diante.

4.3.2.8 Formato 7

Complemento: N > 0, N par
Valor: int12[N] (1 byte mais um nibble)
Tipos associados: sinais representáveis em 12 bits (signed ou unsigned char)
estou correndo Representação no bloco: int12[N]

4.3.2.9 Formato 8

Complemento: N > 0
Valor: uint32[N]
Tipos associados: contadores de 32 bits
Representações especiais: uint32[N], representando o valor dos contadores no início do bloco. uint32[N], representando o valor dos contadores no final do bloco. uint32[N], representando os valores mínimos dos contadores no bloco. uint32[N], representando os valores máximos dos contadores no bloco.
Representação no bloco: uint32[N]

4.3.2.10 Formato 9

Complemento: irrelevante (0)
Valor: pacote de dados DMU (16 bytes)
Tipos associados: valor dos sensores do giroscópio DMU-VGX
Representação no bloco:

```
struct DMU {
        unsigned char angular_rate_x_msb, angular_rate_x_lsb;
        unsigned char angular_rate_y_msb, angular_rate_y_lsb;
        unsigned char angular_rate_z_msb, angular_rate_z_lsb;
        unsigned char acceleration_x_msb, acceleration_x_lsb;
        unsigned char acceleration_y_msb, acceleration_y_lsb;
        unsigned char acceleration_z_msb, acceleration_z_lsb;
        unsigned char temp_sensor_msb, temp_sensor_lsb;
        unsigned char time_msb, time_lsb;
};
```

Capítulo 3 Modificações Desenvolvidas

5.1 Introdução

Na fase inicial deste projeto, o software, apesar de já estar funcionando e apresentando dados, ainda não podia ser utilizado para inspeções, pois os dados apresentados ainda tinham que passar por um processo de correção para que o analista pudesse dizer o que realmente era um defeito. Depois das modificações desenvolvidas nesse projeto pretende-se que o caso de uso mostrado na Figura 3-1 seja possível. Para tanto, três classes foram implementadas: os Condensadores Específicos, os Plug-ins e os Configuradores.

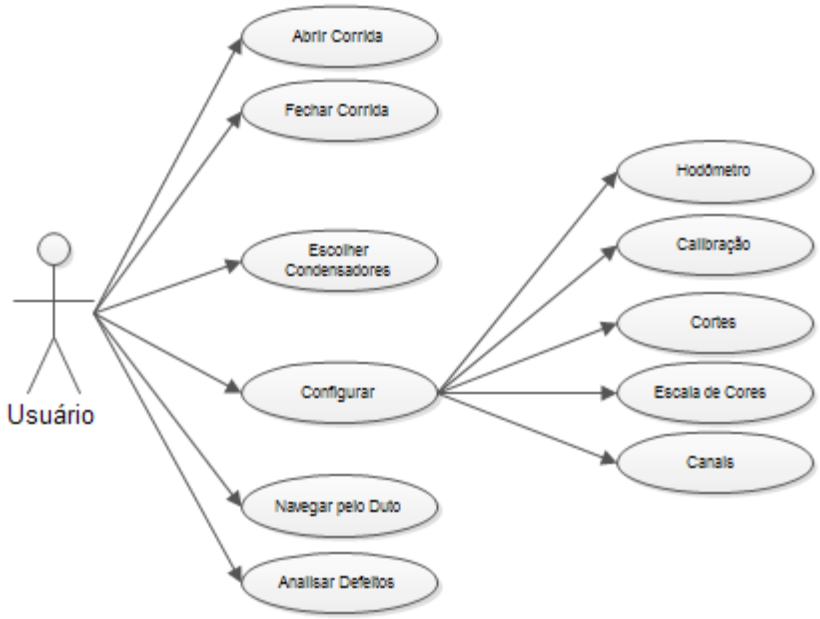

Figura 3-1 Diagrama de Caso de Uso

Como já mencionado, o software só mantém em memória os dados necessários para exibir certa área em tela. Dependendo do nível de zoom, o número de amostras é muitas vezes superior ao número de pixels disponíveis para os gráficos. Para exibi-las, usa-se um Condensador, estrutura que transforma diversas amostras, dependendo da discretização, em uma. No momento essa transformação na verdade é uma escolha: dentre todas as amostras referentes a um pixel, a primeira é a escolhida. No Subcapítulo 4.2, será apresentado o funcionamento dos Condensadores Específicos, condensadores que condensam cada um dos dados nas amostras de forma diferenciada.

Depois de condensadas, as amostras se encontram em um Buffer, e passam por uma cadeia de processamento composta de diversos Plug-ins. Para o RTScan, um Plug-in é um pequeno bloco de processamento. Eles têm essa nomenclatura, pois podemos conectá-los um após o outro de forma a criar uma cadeia de processamento dos dados. Foram desenvolvidos dois tipos de Plug-in: aqueles que processam os dados e aqueles que exibem os dados em uma área de tela. Esses últimos são chamados de Decorators.

Para a otimização de análise de uma corrida, há diversas visualizações que desejamos desenvolver para a secção de duto e cada uma delas será gerada por um Decorator. Muitas dessas visualizações partilham parcialmente do mesmo processamento e, para evitar que dados sejam reprocessados, a cadeia de processamento é estruturada em forma de árvore.

Assim como a cadeia de processamento dos dados, a interface com o usuário, sofreu importantes implementações durante o processo deste trabalho e encontra-se atualmente quase completa, mas em um estado de constante aprimoramento. Os Configuradores, componentes necessários para a modificação das configurações, precisavam também de implementações. Nesses próximos capítulos os Condensadores, os Plug-ins e os Configuradores serão caracterizados e suas implementações, objeto deste trabalho, serão descritas com maiores detalhes.

5.2 Condensadores Específicos

Para explicar a necessidade de condensadores específicos primeiro é preciso explicar como é estruturada uma amostra. Uma amostra vinda de um arquivo blk contém diversos dados, de diversos Formatos. A seguir, um exemplo de um arquivo padrão vindo de um PIG US:

Formato 0 – Mensagens

Geralmente vazio, algumas vezes o PIG gera alguns avisos sobre seus estados.

Formato 3 – Ecos detectados por limiar

Um PIG US adquire centenas de amostras no formato Ascan. Destas amostras ele detecta dois ecos que representam a superfície interna e externa do duto. Há diversos algoritimos de detecção, porém somente dois são implementados na eletrônica embarcada do PIG, principalmente por serem rápidos e simples. Depois de detectados através do algoritimo por limiar, eles são armazenados.

Formato 3 – Ecos detectados por picos de amplitude.

Assim como o dado acima, porém com utilização de um algoritimo diferente.

Formato 4 – Ascan

Guardado a cada N número de amostras para verificar se os algoritimos estão detectando os ecos de maneira eficiente.

Formato 7 – Entradas Analógicas Digitalizadas

Um PIG US conta também com pelo menos dois hodômetros e dois sensores de orientação ligados a uma placa A/D. Como o PIG é impulsionado pelo próprio fluído transportado pelo duto, precisamos ter noção da posição e orientação em que ele se encontra.

Como pode ser visto, cada amostra contém diversos dados de tipos diferentes. Alguns deles são dados quantificados e se pode fazer média, enquanto outros são textos que devem ser somados. Para cada um desses Formatos, um Condensador Específico deve ser criado de forma a satisfazer as necessidades de cada um.

5.2.1 Condensador Padrão

Caso um Formato dentro da amostra não tenha Condensador Específico, esse Condensador é utilizado. Nele seleciona-se, dentre várias amostras, o primeiro dado do Formato que seja não nulo. Caso não haja nenhum dado que se encontre no padrão acima, o dado é marcado como inválido.

5.2.2 Condensador Simples

Esse Condensador apresenta um comportamento muito parecido com o mencionado anteriormente, porém ele seleciona o primeiro dado do Formato que não seja inválido. No RTScan cada Formato contém um espaço reservado para esse valor. A exemplo dos Ecos, eles são guardados em um uint8, e caso o valor seja exatamente igual a 2^8, ele é um dado inválido. Caso não haja nenhum dado que se encontre no padrão acima, o dado é marcado como inválido.

5.2.3 Condensador de Média

Esse Condensador adquire todos os dados do Formato dentre as amostras que não sejam inválidas. Depois de adquiridos, uma média desses valores é feita. Caso não haja nenhum dado que se encontre neste padrão, o dado é marcado como inválido.

5.2.4 Condensador de Mediana

Esse Condensador, apesar de ser nomeado de Mediana, não calcula a mediana e sim algo similar. Quando o usuário ativa esse condensador, ele deve decidir um tamanho X para um armanezador. Para cada N possibilidades de valor, este Condensador cria N/X armazenadores. Cada um desses armazenadores deve receber dados que estejam em suas faixas de valores. Um exemplo, se tivermos dez possibilidades de valor (0-9) e o usuário decide que o tamanho do armazenador é cinco, teremos dois armazenadores. O primeiro recebe os dados entre 0 e 4 enquanto o segundo recebe os dados entre 5 e 9.

Então ele adquire todos os dados válidos do Formato dentre as amostras e os coloca em seus respectivos armanezadores. Ao terminar de associar todos os dados, ele seleciona o armazenador com maior número de dados e uma média dos valores destes dados é feita. Caso não haja nenhum dado que se encontre neste padrão, o dado é marcado como inválido.

5.2.5 Condensador de Máximo

Esse Condensador adquire todos os dados válidos do Formato dentre as amostras. Depois de adquiridos, uma varredura para verificar qual é o maior deles é feita. Caso não haja nenhum dado que se encontre neste padrão, o dado é marcado como inválido.

5.2.6 Condensador de Mínimo

Idêntico ao acima, porém a varredura é feita buscando-se o valor mínimo.

5.3 Configuradores

Os Configuradores são estruturas, com ou sem interface gráfica, que modificam as ConfigKeys. Cada um deles tem a capacidade também de verificar se os valores da chave estão dentro de padrões pré-estabelecidos, e caso esses valores tenham saído dos limites, avisar o usuário e/ou utilizar automaticamente valores padrões.

Alguns desses Configuradores são simples barras de rolagem ou nem têm interface alguma, agindo automaticamente. Enquanto isso, outros envolvem uma caixa de diálogo mais elaborada e exigem valores precisos em milímetros para funcionar corretamente.

Neste capítulo descreveremos cada um deles, ilustraremos sua interface em figuras e a chave que eles modificam. O efeito que cada chave modificada desencadeia nos processos dos Plug-ins é variado e será discutido em capítulos futuros, quando descrevemos os Plug-ins. Aqui só apontaremos os processos que precisam ser refeitos.

5.3.1 Configurador de Físico – Lógico.

Há duas operações de mudança de ordem que necessitam ser feitas nos dados obtidos pelo PIG. A primeira é o mapeamento de canais e a segunda é o ajuste de coroas. A organização dos sensores é feita de forma que os dados, sem um tratamento adequado, quando exibidos em tela, não façam sentido algum. Por tal motivo eles são reordenados e/ou deslocados para que os dados apresentados em tela retratem a superfície do duto.

Essas duas operações são feitas em dois momentos distintos, porém não há sentido em se utilizar uma independentemente da outra. Consequentemente, somente um Configurador foi criado para dar suporte à utilização de seus Plug-ins.

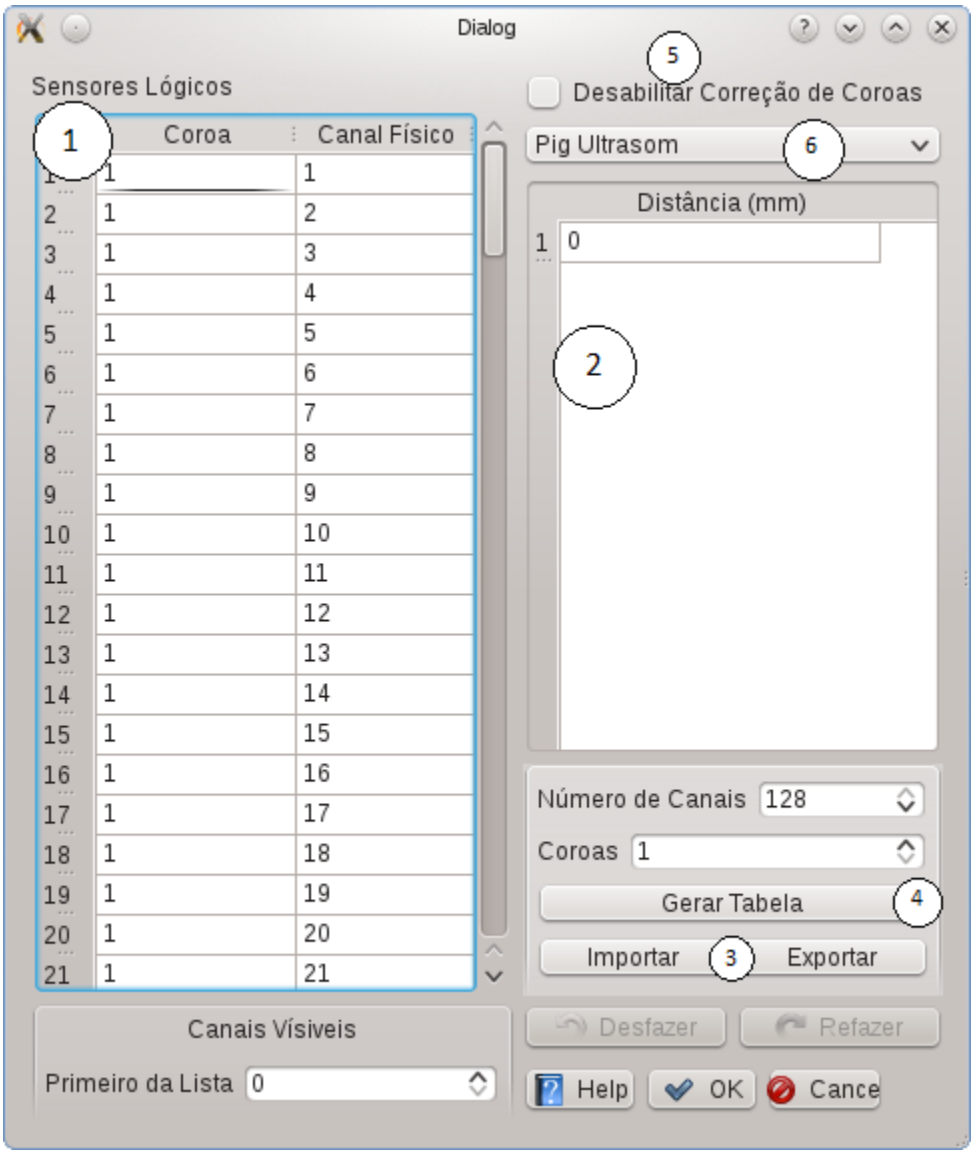

Figura 3-2 Configurador de Físico/Lógico

Como visto na Figura 3-2 o Configurador contém duas tabelas (1 e 2) , seis botões, sendo dois para confirmação (3), dois para importação/exportação das tabelas (4), um para gerar uma tabela padrão (5) e um para habilitar/desabilitar os Plug-in (6), um menu com

opções de padrões para a geração das tabelas (7) e duas caixas de texto para modificar o número de linhas das tabelas (8 e 9).

(1) Tabela de Mapeamento Físico-Lógico– Esta tabela contém informações que serão utilizadas nos dois Plug-ins. Na coluna da esquerda, cada canal lógico é ligado a uma coroa, e, na coluna da direita, a seu respectivo físico. Seu número de linhas é o número desejado de canais lógicos que serão processados nos Plug-ins.

(2) Tabela de Distâncias entre Coroas – Esta tabela tem o número de linhas igual à quantidade de coroas contidas no PIG. É padrão a primeira linha ter o valor 0 e as próximas valores maiores em milímetros que representam a distância que uma coroa tem da outra. Geralmente, em um grupo de coroas, elas têm a mesma distância uma da outra, porém alguns PIGs têm mais de um desses grupos, afastados por distâncias maiores, e por isso se faz válido o uso de uma tabela.

(3) Importar/Exportar – Criar a tabela de mapeamento é um processo trabalhoso e repetitivo, alguns PIGs têm uma quantidade de sensores próxima de 700. Este processo geralmente é feito por somente uma pessoa e depois exportado em um arquivo que é compartilhado entre vários colaboradores.

(4) Gerar Tabela – Este botão gera uma tabela com valores padrão. Há alguns métodos de criação que deveriam ser seguidos também na conexão do conversor A/D, porém isso na maioria das vezes não acontece. Esse botão geralmente serve para uma rápida visualização da corrida, mas para uma vistoria mais detalhada é necessário uma tabela bem mais precisa.

(5) Desabilitar Correção de Coroas - Desabilita a correção de coroas e o mapeamento Lógico-Físico.

(6) ComboBox - Há dois principais padrões, um para PIG Palito e outro para PIG US, mas outros tipos também foram implementados.

5.3.2 Configurador de Calibração

Como fazemos a calibração dos sensores utilizando uma interpolação polinomial de 3º grau, necessitamos de três polinômios para cada um dos canais. Esse Configurador recebe esses valores de duas possíveis maneiras, via importação dos dados ou digitação dos mesmos em uma tabela. Como o arquivo gerado é de acordo com a corrida e a calibração dos sensores é uma característica do PIG, diversas corridas podem se utilizar da mesma calibração, fazendo com que os botões de importar e exportar sejam extremamente úteis.

Esse Configurador pode ser visto na Figura 3-3 e consiste de uma tabela 3xn (1) e um conjunto de botões para importar/exportar (2).

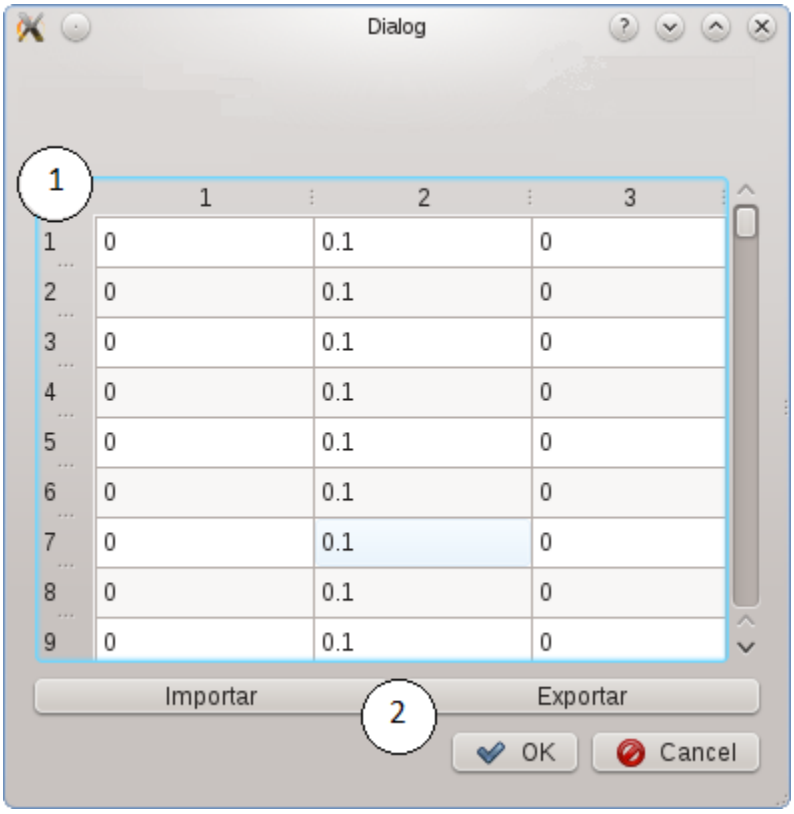

Figura 3-3 Configurador de Calibração

(1) Tabela de Calibração - A tabela consiste dos valores que a calibração necessita para ser feita.

(2) Importar/Exportar - Esses dois botões exportam e importam os dados da tabela. Ao exportar um dado, este é guardado num arquivo de extensão CSV que depois pode ser importado em qualquer corrida.

5.3.3 Configuradores de Janela Principal

Alguns Configuradores são instanciados dentro da própria RTScanMainWindow, que pode ser vista na Figura 3-4. Tais Configuradores não necessitam do Manager para serem inicializados.

5.3.3.1 Configuradores de corte

Este é um Configurador que dá ganho e offset ao gráfico de corte, de forma que seja mais fácil visualizar possíveis defeitos. Há dois gráficos de corte, que representam uma secção longitudinal ou transversal do duto no ponto marcado pelo cursor, e para isso tem-se a necessidade de dois Configuradores: um para o corte lateral e outro para o longitudinal. Eles contêm um botão (1) e duas barras deslizantes (2 e 3).

(1) Botão que centraliza gráfico, e impossibilita a utilização da barra (2). (chave: AutoOffsetEnabled)

(2) Barra que ajusta o offset do gráfico (chave: OffSet)

(3) Barra que ajusta o ganho do gráfico (chavo: Scale)

5.3.3.2 Configurador de Posição Horária

Esse Configurador foi feito imitando o comportamento de uma barra de rolagem, porém com algumas funções adicionadas à mesma. Além do uso comum de excursionar através da chapa, ela também é capaz de modificar o zoom e mostrar os canais que estão em tela e o número total de canais que estão sendo processados.

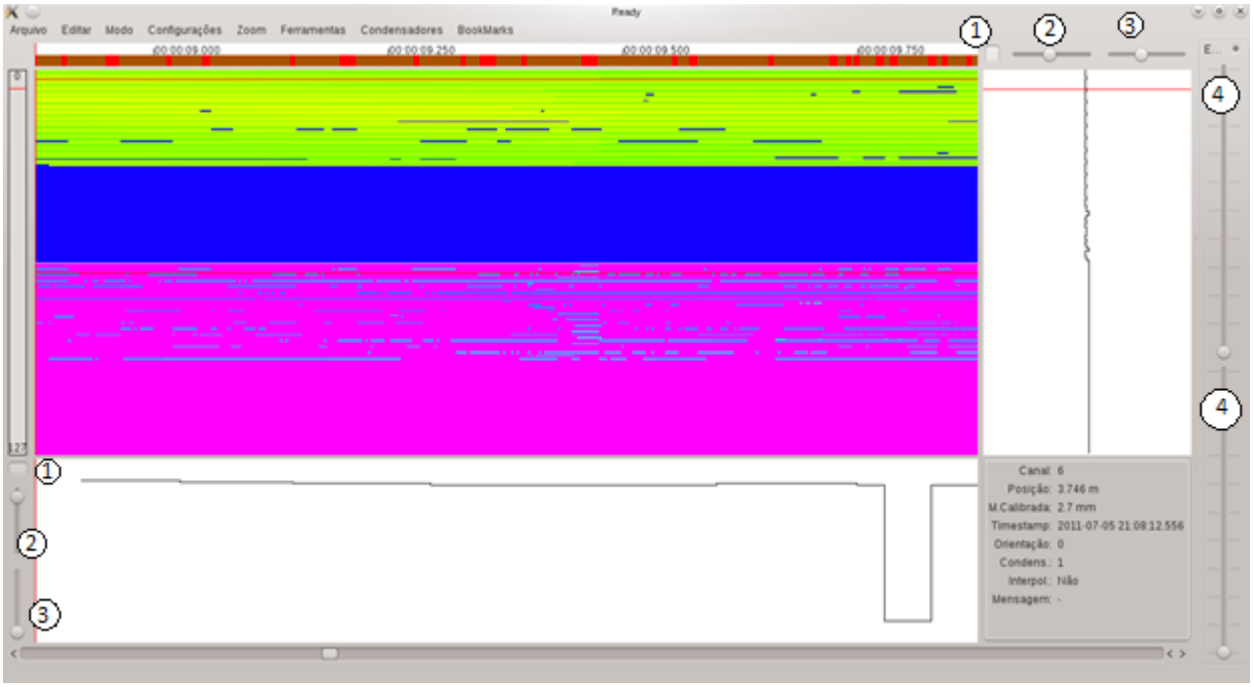

Figura 3-4 RTScan-Janela Principal

O Configurador é composto por apenas uma barra, porém ela tem três possíveis pontos de interação e duas informações extras. Suas funções são:

- Faz com que a barra suba ou desça um passo de tela. (chave: FirstChannelInScreen)
- Mouse segura barra e ela pode deslizar na direção desejada.(chave: FirstChannelInScreen)
- Barra pode ser esticada fazendo com que mais ou menos canais sejam apresentados em tela (chave:channelsShown)
- Limites totais sendo processados pelos Plug-ins
- Limites parciais sendo mostrados em tela.

5.3.3.3 Configurador de Escala de Cores

Assim como os Configuradores de corte, esse Configurador serve para dar mais destaque aos defeitos. Ele controla o nível de variação das cores e o deslocamento da área trabalhada. Utilizamos um sistema de cores chamado HSV, (*hue*, *saturation*, *value*). Esse sistema de cores define o espaço de cor conforme descrito abaixo, utilizando seus três parâmetros:

- Matiz (tonalidade): Verifica o tipo de cor, abrangendo todas as cores do espectro, desde o vermelho até o violeta, mais o magenta. Atinge valores de 0 a 360º, mas na

nossa aplicação, esse valor é normalizado de 0 a 100% e utilizamos somente 80% desses valores, impossibilitando que a mesma cor seja a simbolização do valor mínimo e máximo.

- Saturação: Também chamado de "pureza". Quanto menor esse valor, mais com tom de cinza aparecerá na imagem. Atinge valores de 0 a 100%.
- Valor (brilho): Define o brilho da cor. Atinge valores de 0 a 100%.

No RTScan há dois tipos de escalas de cores, uma delas somente em escalas de cinza e a outra que varia entre o vermelho e o violeta.

Escala de Cinza:

Nessa escala a Saturação é fixada em 0% e as barras variam o brilho. Nesse caso a Matiz não tem importância. Quando a saturação está em 0% ela perde completamente a necessidade de ser variada.

Escala de Cores:

Nessa escala a Saturação é fixada em 100% e o brilho em 100%; as barras variam a Matiz.

Temos duas barras uma que modifica um offset e outra que modifica um ganho que podem ser vistas na Figura 4-4 demarcadas com o número quatro.

5.3.4 Configuradores sem UI

Alguns Configuradores não têm uma interface. Há a necessidade desses Configuradores existirem, porque algumas chaves são limitadas por outras chaves ou até mesmo modificadas em consequência da alteração do valor de um conjunto de chaves.

5.3.4.1 Configurador de Distância em Pixels

O propósito desse Configurador é calcular a quantidade de unidades de tela que o Plug–in de ajuste de coroas necessitará. Ele depende de diversas chaves: Distâncias em mm, Discretização, Pulsos de Hodômetro por Rotação, Diâmetro do Hodômetro em mm.

5.3.4.2 Configurador de Zoom Vertical

O propósito deste Configurador é impedir que certas chaves ultrapassassem limites variáveis. Como o zoom vertical depende de duas chaves, Primeiro canal em tela e Número de canais em tela, não se pode deixar que três coisas aconteçam:

- O somatório do primeiro canal em tela e o número de canais seja maior que o total de canais.
- O primeiro canal seja menor que o limite inferior de canais.
- O canal selecionado esteja contido na tela.

Este Configurador dá prioridade à chave que foi modificada mais recentemente. Dessa forma, se o canal selecionado for menor que o primeiro canal em tela, este diminue de forma que o canal selecionado volte a aparecer em tela.

5.4 Plug-ins

Plug-ins são blocos para modificação do dado a ser mostrado em tela. Cada bloco é uma operação necessária, podendo: particionar os dados de entrada em vários tipos de dados, transformá-los, exibi-los ou simplesmente deslocar os dados de um buffer. Como utilizamos uma divisão de trabalho com múltiplos threads, esses blocos devem ser paralelisáveis de forma a aumentar o desempenho do programa.

Esse bloco pode receber um número ilimitado de entradas e gerar um número ilimitado de saídas, sendo estas definidas em um arquivo XML, onde é anotada cada uma das propriedades dos Plug-ins e das suas devidas entradas e saídas. Há dois tipos de Plug-ins, os de uma etapa e aqueles de duas etapas, sendo o segundo um caso especial do primeiro. Todo Plug-in é uma classe derivada de IPlugin que implementa dois métodos virtuais, process e setupChannels. No caso dos Plug-ins de duas etapas há dois processos sendo o segundo chamado de processSecondStep.

O método setupChannel recebe a estrutura dos canais dos dados de entrada e devolve essa estrutura como ela irá ficar após o processamento da informação. O método Process é onde propriamente acontecerá o processamento. Ele recebe uma estrutura chamada CondensedSampleBuffer, um vetor de índices de entrada, um vetor de índices de saída e a extensão de amostras que devem ser processadas por esse Plug-in. O CondensedSampleBuffer, como o próprio nome diz, é um buffer de amostras já condensadas. Antes do processamento de todos os Plug-ins, o Manager aloca uma nova área de memória para esse buffer e entrega ao processamento um índice relacionado a esta área como saída. Esta é preenchida e, depois que o processamento termina, o índice é entregue como entrada de um próximo Plug-in.

Quem determina as dependências das entradas e saídas de cada Plug-in é o Resolvedor, uma classe que gera a árvore de Plug-ins. A árvore é composta de ramos cujas folhas são Decorators, uma denominação para os Plug-ins que exibem os dados em tela e a raiz é o dado vindo do condensador.

O próximo capítulo irá ilustrar essa árvore de Plug-ins e descrever a implementação de alguns deles. Além disso, apontaremos quais ConfigKeys são necessárias para cada um dos Plug-ins e apontaremos que algumas chaves podem até mesmo retirar um Plug-in da árvore.

5.4.1 Plug-in Tuple-Array

A eletrônica do PIG já mudou muito durante os anos. No passado ela detectava os ecos do pulso de ultrassom e os guardava no arquivo. Como era comum que o algoritimo de detecção não conseguisse capturar com perfeição tais ecos, quando o analista fazia a inspeção alguns dados se apresentavam incongruentes.

No presente momento um novo tipo de PIG está sendo desenvolvido. Nele toda a resposta ao pulso de Ultrassom é guardado no arquivo blk. Para isso ele é guardado em um Tuple-Array, estrutura que permite guardar vetores compactados de diversas formas. Esse Plug-in é capaz de detectar qual a compactação presente no Tuple-Array e após descompactá-lo, guardar os dados no buffer saída.

5.4.2 Plug-in de Detecção de Picos por Limiar

Depois de descompactado, cada canal de cada amostra tem um vetor com a resposta a um pulso de US. Nesse Plug-in utilizamos um algoritimo para detecção de picos neste vetor. Antes de apresentar o algoritmo separadamente, é necessário definir alguns parâmetros básicos, identificados na **Figura** 3-5.

- dtime1 - Instante de tempo contado do início da aquisição, a partir do qual será feita a busca do primeiro pico, e que corresponde à interface fluído-metal. Está associado ao delay de aquisição e à distância do cabeçote à parede interna do duto.
- w1 – Corresponde ao tamanho da janela sobre a qual será feita a detecção do primeiro pico.

- dtime2 – Intervalo de tempo contado a partir do ponto em que foi detectado o primeiro pico, indicando o ponto em que se inicia a busca dos segundos picos (picos com características semelhantes que são subsequentes ao primeiro). O dtime2 também representa a espessura mínima da parede de metal que poderá ser medida e pode ser chamado de "zona morta" a partir da superfície do metal.
- w2 - Corresponde ao tamanho da janela sobre a qual será feita a detecção dos segundos picos. Também está associado à espessura máxima da parede do duto que pode ser medida.

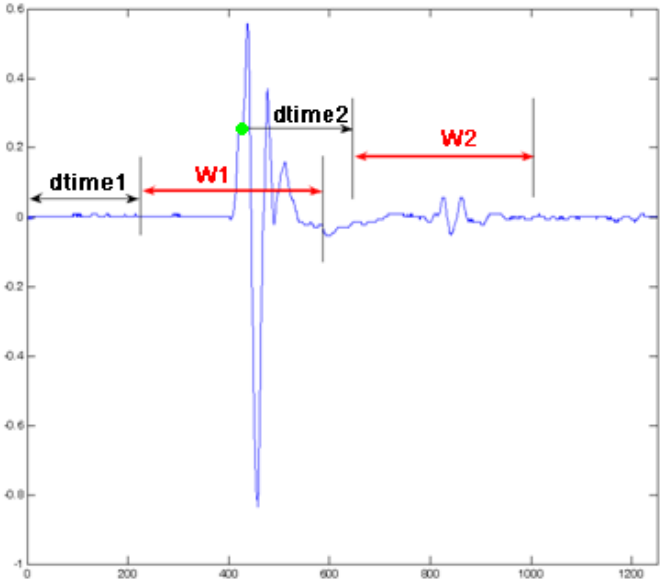

Figura 3-5 Parâmetros básicos da detecção de picos. (gráfico: amplitude versus tempo)

5.4.2.1 Detecção por limiar estático

A detecção de echos por limiar estático é um método bastante simples que realiza a detecção dos picos a partir de um detector de nível no sinal analógico. Consiste basicamente em procurar, dentro das janelas w1 e w2, os pontos em que o valor do sinal ultrapassa um determinado valor de tensão pré-estabelecido. Esse método tem a grande vantagem de realizar a detecção em tempo real, ou seja, a detecção pode ser feita paralelamente à aquisição dos dados sem inserir atrasos no processo de medição, o que torna o método extremamente rápido e reduz o uso de memória.

Devido à sua simplicidade e rapidez, o método em questão pode ser empregado também em paralelo com os demais algoritmos com a função de validar a detecção dos picos de ultra-som de acordo com o seguinte critério: se a amplitude do sinal na amostra identificada como um pico estiver abaixo do limiar, essa medição é descartada. Esse algoritmo se mostra um eficiente critério de validação quando se utilizam valores de limiar diferentes para o primeiro e os segundos picos.

Outra vantagem desse algoritmo é detectar os picos no seu primeiro ciclo, o que lhe garante certa imunidade sobre os efeitos de alargamento de pico. No entanto, esse algoritmo apresenta uma grande dificuldade relacionada à escolha do valor do limiar, pois, quando este valor é sub-dimensionado, ruídos presentes no sinal são detectados como falsos picos e, quando ele é super-dimensionado, picos muito atenuados passam despercebidos, o que ocorre

com freqüência em regiões com corrosão. Para diminuir este efeito no RTScan, utilizamos dois limiares, um para detecção do primeiro pico e outro para a detecção dos segundos.

Na Figura 3-6 podemos ver um exemplo onde o limiar é o mesmo para todos os picos e vemos que ele consegue identificar três picos, apesar da existência de outros mais adiante.

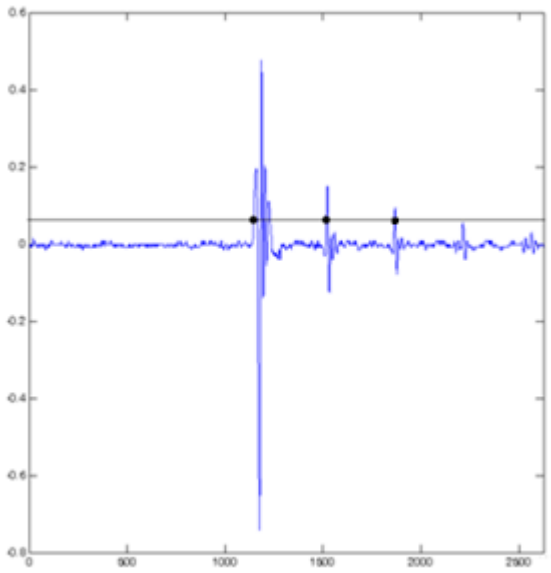

Figura 3-6 Detecção por limiar. Os pontos indicam os picos adquiridos. (gráfico: amplitude versus tempo)

Depois de identificados, os índices dos picos são guardados em um vetor, que segue no buffer de saída. Multiplicando esses índices pela frequência de aquisição do sensor ultrossônico podemos identificar o momento em que ocorreu determinado pico.

5.4.3 Plug-in Seletor de Picos

Esse Plug-in seleciona dentre os picos os melhores para representar os dois tempos desejados para análise. Estes tempos são denominados ecos; um é utilizado para a determinação da espessura da coluna de fluído e o outro para a espessura do metal, gerando assim duas saídas.

Em mais de 98% das aquisições, o tempo de aquisição do primeiro pico mais o deadzone do sensor é o que melhor representa o primeiro eco, tempo que a onda leva para ir até a superfície interna do duto e voltar. Isso só não é verdadeiro quando há um ruído no sinal; futuramente espera-se que um filtro possa retirar estes ruídos.

Há mais de uma forma de se determinar o segundo eco, tempo que o som leva para ir e voltar dentro da camada de metal. Como geralmente acontencem mais de uma reflexão dentro da camada de metal do duto, procura-se o tempo entre o (n-1)-gésimo e n-gésimo pico. Como algumas vezes captura-se um ruído como pico, há a necessidade de se verificar a similaridade com o tempo entre o primeiro e o segundo pico. Caso estes não sejam similares, compara-se o par de picos anterior até que um par seja eleito. Caso menos de dois picos sejam selecionados, o segundo eco é marcado como inválido, e caso menos de um pico seja selecionado, o primeiro eco é marcado como inválido.

5.4.4 Plug-in de Interpolação

5.4.4.1 Interpolação

No campo matemático da análise numérica, interpolação é um método de construção de novos pontos de dados dentro do intervalo de um conjunto discreto de pontos de dados conhecidos. Ela é muitas vezes necessária para estimar o valor intermediário de um número de pontos de dados, obtidos por amostragem ou experimentação, que representam os valores de uma função para um número limitado de valores da variável independente. Isto pode ser conseguido por ajuste da curva ou análise de regressão.

Um problema diferente que está estreitamente relacionado com a interpolação é a aproximação de uma função complicada por uma função simples. Suponha que sabemos a fórmula de uma função, mas ela é muito complexa para ser avaliada de forma eficiente. Então nós poderíamos escolher alguns pontos conhecidos de dados da função complicada, criando uma tabela de pesquisa e tentar interpolar os pontos dados para a construção de uma função mais simples. Claro que, ao usar a função simples para estimar novos pontos, geralmente não produzimos o mesmo resultado que teríamos se tivéssemos usado a função original, mas dependendo do domínio do problema e do método de interpolação utilizado, o ganho de simplicidade pode compensar o erro.

5.4.4.1.1 Interpolação por partes constantes

O método mais simples de interpolação é localizar o valor mais próximo dos dados e atribuir o mesmo valor. Um exemplo pode ser visto na Figura 3-7. Esta é uma escolha favorável por sua velocidade e simplicidade.

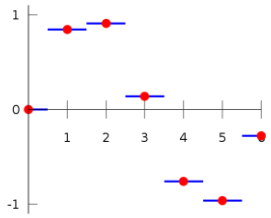

Figura 3-7 Interpolação por partes constantes

5.4.4.1.2 Interpolação Linear

Interpolação linear é rápida e fácil, mas não é muito precisa. Para dois pontos dados, a interpolação linear é a linha reta entre esses pontos. Uma desvantagem é que a interpolação não é diferenciável nos pontos, como pode ser visto na Figura 3-8.

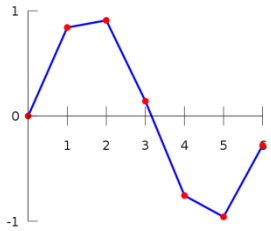

Figura 3-8 Interpolação linear

5.4.4.1.3 Interpolação Polinomial

Interpolação polinomial é uma generalização da interpolação linear. Note-se que a interpolação linear é uma função linear. Vamos agora substituir esta interpolação por um polinômio de grau mais elevado. Geralmente, se temos n pontos de dados, então haverá um polinômio de grau no máximo n-1 passando por todos os pontos de dados. A interpolação é um polinômio e, portanto, infinitamente diferenciável. Assim, vemos que a interpolação polinomial supera o principal problema da interpolação linear. No entanto, interpolação polinomial também tem algumas desvantagens. O cálculo do polinômio interpolador é computacionalmente caro em comparação com o da interpolação linear. Além disso, interpolação polinomial pode apresentar grandes oscilações, especialmente nos pontos finais.

5.4.4.1.4 Interpolação por Spline

Interpolação por Spline usa polinômios de baixo grau em cada um dos intervalos e escolhe os polinômios tais que os intervalos se encaixem de forma que exista derivada nos pontos. A função resultante é uma Spline. Como a interpolação polinomial, a interpolação por Spline incorre em um erro menor do que a interpolação linear. No entanto, a linearização é mais fácil de avaliar do que os polinômios de alto grau utilizados na interpolação polinomial, gerando menor gasto computacional.

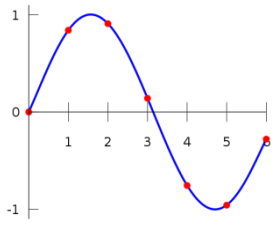

Figura 3-9 Interpolação Polinomial e por Spline

Apesar das interpolações polinomiais e por Spline gerarem funções mais suaves, como pode ser visto na Figura 3-9, a quantidade de dados numa corrida pode fazer com que o computador gasta muito tempo computando uma interpolação polinomial. Apesar da interpolação por Spline diminuir muito esse tempo, os dados obtidos numa corrida tem momentos sem variação alguma e, em alguns momentos específicos, sofrem grandes variações. Isso faz com que a função gerada seja extremamente oscilatória.

5.4.4.2 Conclusão:

Apesar de a interpolação linear parecer fazer mais sentido, ela nem sempre é válida (no caso de Ascans, por exemplo). Por tal motivo haverá a possibilidade de escolher entre a linear e a constante por partes, sendo a segunda o padrão.

5.4.4.3 Implementação:

Esse Plug-in tem limites de dados estendidos de forma que ele recebe mais dados que os que vão para a tela. No início do processo ele busca por uma amostra, dentro do limite

estendido, e a armazena para que seja capaz de preencher um possível espaço vazio no início da tela. Depois, ele analisa um grupo de amostras. Cada uma das amostras que tenha dados, é copiada para o SampleBuffer de saída e substitui a amostra arquivada internamente. Caso a amostra em análise não tenha dados, a amostra arquivada é copiada e preenche o lugar vazio recebendo uma flag, que indica que ela não passa de uma interpolação.

5.4.5 Plug-in de Mapeamento Físico-Lógico

O PIG, como já dito anteriormente, contém um conversor A/D e diversos sensores. Porém, tais sensores são ligados ao conversor em ordem de proximidade. Isso, apesar de diminuir a quantidade de cabos na eletrônica, faz com que a ordem de conexão no conversor e sua posição horária não tenham necessariamente qualquer ligação. Como todas as apresentações gráficas de uma corrida são representações de um duto, há a necessidade dos canais serem apresentados de acordo com sua posição horária. Além disso, como a montagem de cada PIG é diferente, há variações como: o número de canais físicos, o número de hodômetros e de sensores com outras finalidades e, principalmente, a ordem em que os sensores de medição são associados à placa conversora A/D. De forma a resolver tais problemas, foi criado um mapa de correlação entre canais físicos e canais lógicos em forma de tabela. Outra utilidade para esse mapeamento é a repetição de um canal. Algumas vezes um sensor quebra durante a corrida, seja por um defeito que o atingiu ou por um problema no próprio sensor. Quando isso acontece, se não houver como preencher essa falta de dados, o gráfico fica com uma grande área sem dados. Como há diversos sensores com uma posição próxima à do quebrado, o sensor defeituoso deixa de ser considerado no gráfico e substituído durante o mapeamento.

Na Figura 3-10 temos um exemplo de um mapeamento sendo executado. Cada um dos canais é representado por uma cor. Neste exemplo, o canal 0 (laranja), na verdade, pertence à posição 2; o canal 1 (amarelo) quebrou de forma que será substituído pelo canal 3; o canal 2 (azul) pertence à posição 3; o canal 3 (roxo) à posição 4 e, por último, o canal 4 (verde) pertence à posição 0.

Figura 3-10 Mapeamento Físico-Lógico

Para a produção da tabela, um Configurador é criado. Este Configurador contém quatro itens: uma tabela, que é utilizada tanto para o Plug-in de Mapeamento Físico-Lógico quanto para o de Ajuste de Coroas, um botão para importar, um botão para exportar essa

tabela de/para um arquivo e um botão para desativar tanto o mapeamento de canais quanto o ajuste de coroas.

A tabela é simples, assim como a implementação do código. Um loop percorre cada posição da tabela verificando qual canal físico deve ser copiado para uma nova amostra somente com canais lógicos. Essa tabela é de tamanho limitado de forma que o número de canais lógicos nunca possa ser maior que o de canais físicos. Ela também contém valores limitados garantindo que não se possa fazer a associação de um canal físico inexistente.

5.4.6 Plug-in de Ajuste de Coroas

Como já mencionado, todo o processo dos Plug-ins guarda em memória somente dados suficientes para exibir a área exata da tela. Para isso, geralmente os Plug-ins recebem uma área num buffer já alocado onde vão trabalhar e uma área de tamanho idêntico com dados. Alguns, porém, precisam de limites estendidos de dados de entrada. O Plug-in de Coroas é um desses casos. Como há o deslocamento horizontal de canais, eles precisam de buffers de entrada maiores que os de saída. Essa informação sobre o tamanho dos buffers é retirada de umas das chaves e calculada antes dos processamentos dos Plug-ins acontecerem. Dessa forma todos os Plug-ins anteriores a ele sabem que têm que gerar uma quantidade de dados suficiente para preencher a tela de visualização.

O Plug-in de Ajuste de Coroas depende somente de duas chaves, uma que guarda à qual coroa um canal pertence e uma que guarda as distâncias, em unidades de tela, que cada conjunto de canais pertencentes à mesma coroa, deve ser deslocado.

O processo é bem simples, um loop duplo varre tanto as amostras quanto os canais de cada amostra e posiciona o dado numa amostra mais à esquerda, porém no mesmo canal. Como cada coroa tem um deslocamento diferente, dependendo de à qual coroa esse canal pertença, diferente será o deslocamento de amostras que o dado sofrerá.

No exemplo da Figura 3-11 temos somente 4 canais e cada um deles pertence a uma coroa diferente. No primeiro quadro temos o buffer de entrada onde a área necessaria para a criação de uma tela é representada pelo retângulo com espessura maior. Durante o processamento cada coroa é deslocada em uma certa distância e por tal motivo as áreas em vermelho são descartadas e não aparecem em tela. No segundo quadro podemos ver a disposição dos dados após o processamento.

Figura 3-11 Ajuste de Coroas

5.4.7 Plug-in de Escala e Offset

Esse Plug-in atua nos dados que vão para os dois cortes. Como os dados podem ser guardados em diversos tipos (u_int8, u_int12 ou u_int16), quando plotados em um gráfico,

algumas variações podem ser imperceptíveis. Para permitir a visualização de tais variações, uma escala e um offset, contidos em duas chaves, são aplicados. Se a chave AutoOffsetEnabled estiver ativada, esse Plug-in conta ainda com a possibilidade de determinar um offset ideal para o trecho de duto. Ao detectar os valores máximo e mínimo da área em tela, ele coloca a média desses dois valores centralizada em tela.

5.4.8 Decorators

Este tipo especial de Plug-in diferencia-se principalmente pelo fato da saída dele não ser em forma de dados em buffer e sim de dados visuais. Além de todos os métodos já descritos no subitem 4.4, esta classe tem também um método update, através do qual ele recebe uma referência a uma área em tela. Cada uma destas áreas está numerada na Figura 4-12.

5.4.8.1 Decorator Chapa

Recebe do buffer de entrada cores em RGB e cria um quadro de cores onde cada pixel representa um canal de cada uma das amostras em tela. Depois de escalonado para o tamanho da tela, o quadro é exibido na área de tela principal do programa.

5.4.8.2 Decorator Corte

Processa alturas em pontos que são plotados nas áreas de corte do software.

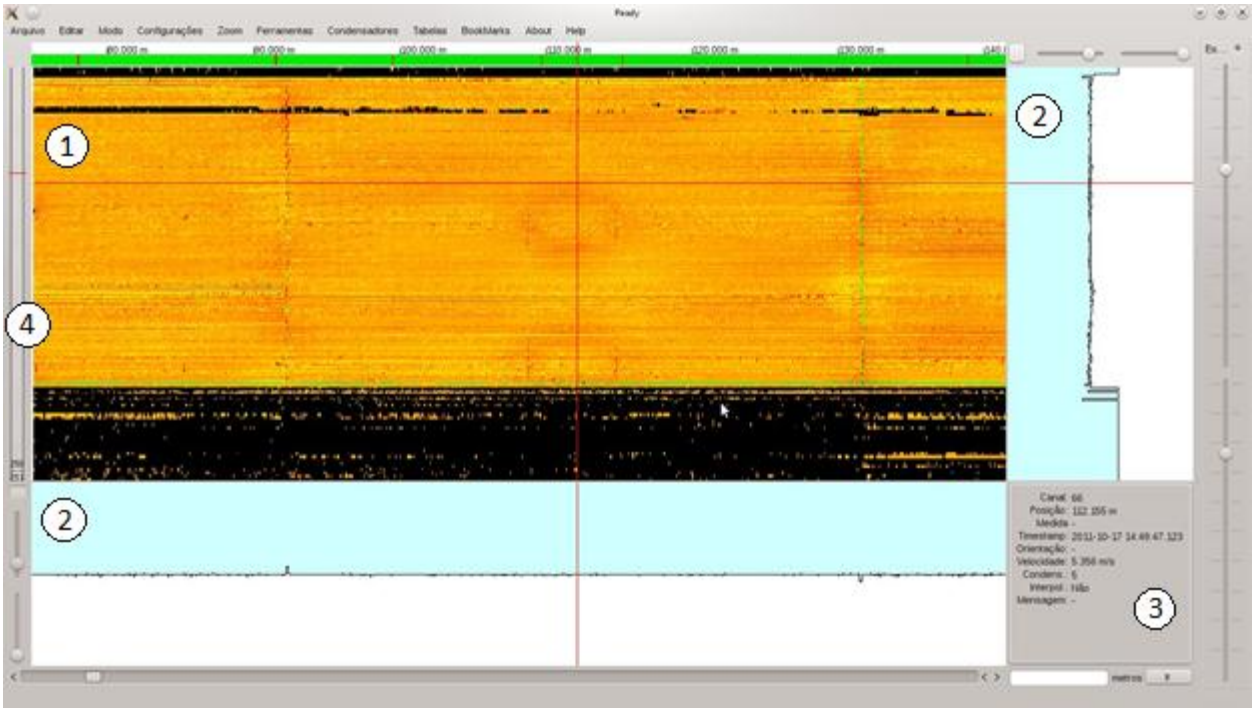

Figura 4-12 Tela Principal do RTScan Atual;
1 – Chapa; 2 – Cortes; 3 – InfoAmostra;
4 – Barra de Posição Horária;

5.4.8.3 Decorator InfoAmostra

Este Decorator recebe duas entradas. Estas duas entradas referem-se à amostra selecionada através da "mosca", porém diferenciam-se, em uma das entradas ela está calibrada e na outra não. Este Decorator exibe as características da amostra e do canal selecionado, essas informações são, canal lógico, posição em m, medida calibrada em mm, TS em formato de data, orientação em horas, velocidade em m/s, número de amostras condensadas, flag interpolada, mensagens oriundas do PIG, canal físico, posição em tics de hodômetro, medida em medidas de A/D, TS em ticks, orientação em medidas de A/D, velocidade em tics de hodômetro/ tics de TS.

5.4.8.4 Decorator de Posição Horária

Este Decorator não recebe nenhuma informação via buffer. Todas as informações nele mostradas são transmitidas a ele através do ConfigKeys. Através dele somos capazes de ver o configurador de Posição Horária, já mencionado no subitem 4.3.3.2.

Capítulo 4 Conclusão

Como conclusão deste trabalho apresentaremos uma comparação entre o RTScan original e o software atual após as implementações. Abaixo podemos ver quatro figuras, em todas elas a mesma corrida está aberta, numa posição parecida. As Figuras 5-1 e 5-2 apresentam o software original. O gráfico na Figura 5-2 apresenta um Zoom centrado em um ponto da Figura 5-1. Em comparação, as Figuras 5-3 e 5-4 apresentam o software depois de todas as implementações discutidas durante este projeto. E assim como as figuras anteriores, a segunda apresenta um Zoom centrado em um ponto da primeira. Para facilitar a comparação, algumas particularidades deste trecho do duto que podem ser notadas nas figuras serão apresentadas a seguir. Como elementos comparativos, enumeramos:

Derivação: Uma derivação é um novo duto que se conecta ao duto vistoriado. A principal característica é a grade que impede que objetos passem de um duto para o outro.

2. Curva: Uma curva gera um efeito curioso nos sensores, como pode ser visto nas figuras abaixo. Como o PIG tem poucos pontos de flexão, no momento da curva, um dos lados dele fica bem mais próximo da parede do duto, enquanto o outro fica mais distante. Desta forma, os sensores da parede externa da curva perdem o sinal e os da parede interna os captam com maior intensidade.

3. Solda: Uma solda consiste do ponto de ligação entre dois canos. Essas soldas são facilmente detectadas, pois provocam uma grande variação no sinal do sensor.

4. Defeito: Geralmente um defeito aparece como uma mancha nos gráficos, sendo extremamente difícil percebe-los.

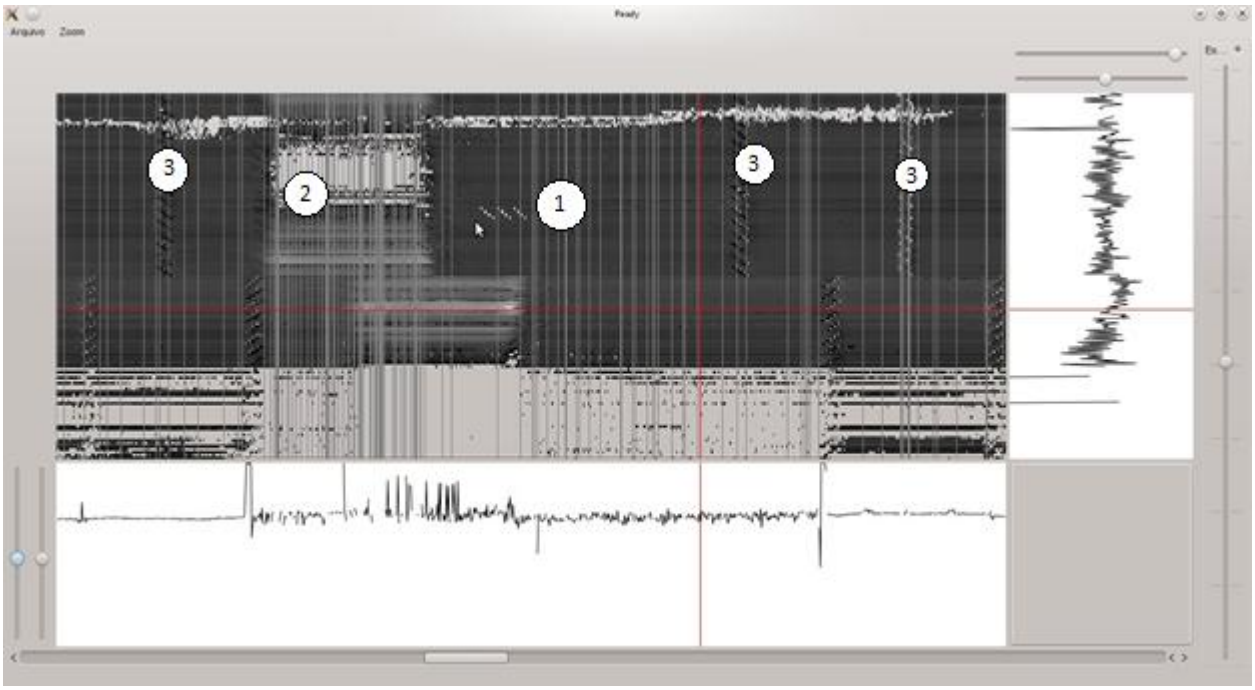

Figura 5-1 Tela do RTScan Original com elementos comparativos: 1 – Derivação; 2 – Curva e 3 – Solda;

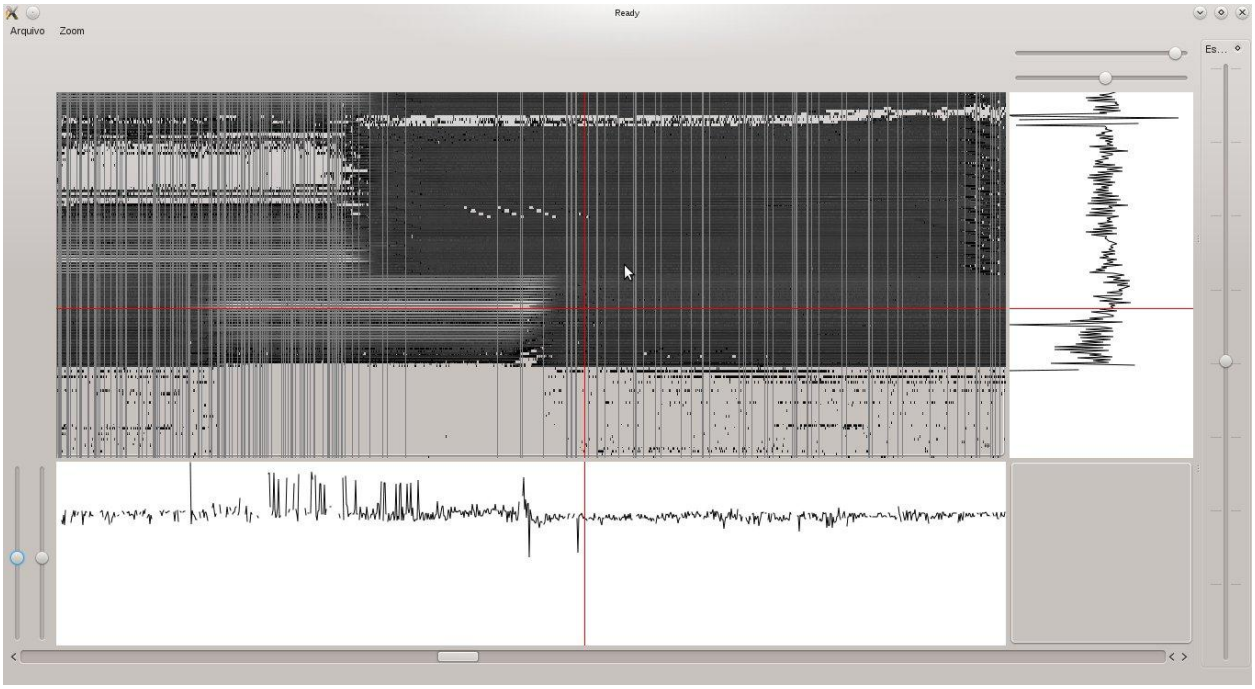

Figura 5-2 Zoom centrado na derivação da Figura 5-1 utilizando o RTScan Original.

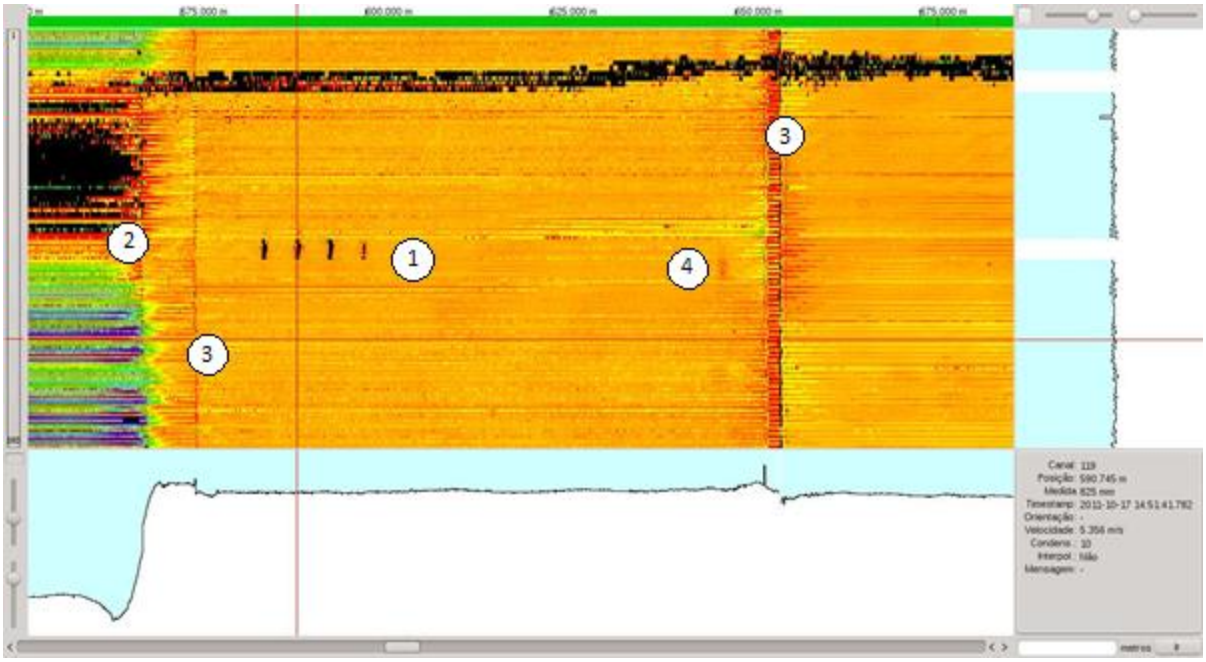

Figura 5-3 Tela do RTScan Atual com elementos comparativos: 1 – Derivação; 2 – Curva ; 3 – Solda e 4 – Defeito.

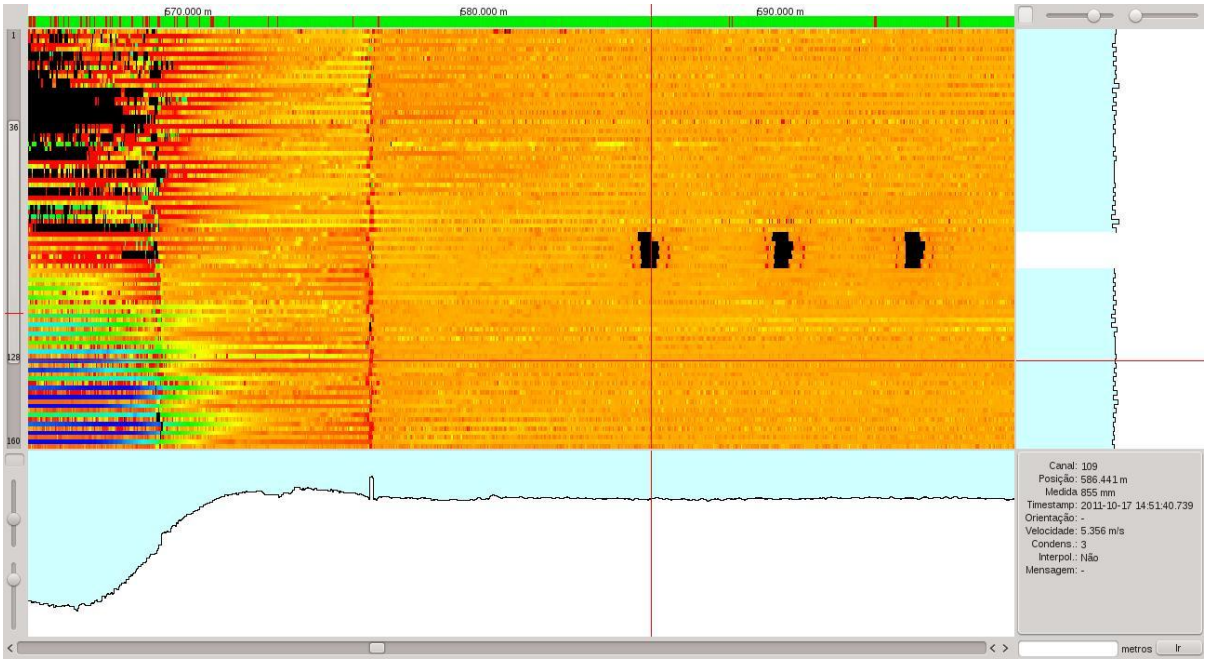

Figura 5-4 Zoom centrado na derivação mostrada na Figura 5-3 utilizando o RTScan Atual.

Como podemos notar, pela análise das figuras acima, houve uma grande evolução na parte visual dos gráficos após as implementações-tema deste trabalho. Como exemplos, podemos citar:

- A visualização gráfica de uma derivação (1), que nas Figuras 5-1 e 5-2, encontra-se diagonalizada; a de uma curva (3) e a de uma solda (2), que se encontram em zig-zag, não retratam a estrutura física do equipamento. Por outro lado, pela utilização do Plug-in de Ajuste de Coroas, nas Figuras 5-3 e 5-4, estas visualizações que representam tais eventos se encontram em forma retilínea.
- Outra implementação importante é a possibilidade de verificarmos a posição, orientação e o raio interno do duto, através do InfoAmostra.
- Com a mudança da escala de cores da chapa, os defeitos (4) ficam bem mais perceptíveis. Como a variação gerada por um defeito é pequena, numa escala com pouca variação de cores a dificuldade de encontra-lo era enorme.
- Na parte inferior das Figuras 5-1 e 5-2, observa-se uma área cinza, que ocupa quase um quarto da tela, representando dados perdidos, pois um grupo de sensores foi danificado. Com a utilização do Mapamento de Canais Físico-Lógico podemos substituir esses sensores por alguns próximos e ver melhor as áreas que nos interessam.
- Também podem ser vistas algumas linhas verticais brancas nos gráficos das Figuras 5-1 e 5-2, que representam amostras perdidas. Com o uso do Plug-in de Interpolação essas amostras são substituídas pelas anteriores de forma a evitar tais falhas no gráfico.
- A Interpolação em conjunto com a nova possibilidade de se fazer zoom verticalmente possibilita um zoom muito mais poderoso fazendo com que detalhes ganhem realce. Um exemplo é a derivação na Figura 5-4.

Após as implementações apresentadas neste trabalho, a nova versão do software RTScan passou a ser utilizada nas análises dos dutos, mas ainda somente como visualizador dos dados obtidos em uma corrida. Por ser bem mais rápida ao fazer o processamento dos dados (as representações dos dutos são geradas em até um décimo do tempo que a versão

anterior necessitava), os analistas a utilizam para detectar defeitos. Após a detecção de um evento, eles trocam para a versão anterior do software para marcar tais eventos, que são guardados em um banco de dados, ainda inexistente na versão atual. Esta marcação de defeitos e o banco de dados são as próximas implementações que irão permitir que a nova versão do RTScan substitua a sua versão anterior no momento da análise de dutos.

Bibligrafia

CAMERINI, C.; BENINCAZA, H.; CASTRO, R. ***Medição de espessura com ultra-som****:* tecnicas tradicionais e avançadas. Rio de Janeiro: Centro de Pesquisa em Tecnologia de Inspeção, 2011.

CENTRO DE PESQUISA EM TECNOLOGIA DE INSPEÇÃO. ***Especificação de formato de dados de corrida de PIG***. v. 1.11, Rio de Janeiro, 2012.

HSL and HSV. Wikipedia. Disponível em: <http://en.wikipedia.org/wiki/HSL_and_HSV>. Acesso em: 12 ago. 2012.

Interpolation. Wikipedia. Disponível em: <http://en.wikipedia.org/wiki/Interpolation>. Acesso em: 10 jun. 2012.

SALCEDO, T. ***Análise do sensor de um PIG instrumentado do tipo palito*** Disponível em: <http://www.maxwell.lambda.ele.puc-rio.br/Busca_etds.php?strSecao=resultado&nrSeq=15689@1>. Acesso em: 06 dez. 2012.

Qt Reference Documentation Disponível em: < http://doc.qt.digia.com/qt/>. Acesso em 25 out 2011.