UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA MECÂNICA

SINCRONIZAÇÃO, CAPTURA E ANÁLISE DE IMAGENS DA POÇA DE SOLDAGEM NO PROCESSO GMAW CONVENCIONAL, NO MODO DE TRANSFERÊNCIA METÁLICA POR CURTO-CIRCUITO

LUCIANO DUARTE NASCIMENTO FRANCO

ORIENTADOR: GUILHERME CARIBÉ DE CARVALHO

DISSERTAÇÃO DE MESTRADO
EM SISTEMAS MECATRÔNICOS

PUBLICAÇÃO:

BRASÍLIA/DF: 17 DEZEMBRO - 2007.

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA MECÂNICA

SINCRONIZAÇÃO, CAPTURA E ANÁLISE DE IMAGENS DA POÇA DE SOLDAGEM NO PROCESSO GMAW CONVENCIONAL, NO MODO DE TRANSFERÊNCIA METÁLICA POR CURTO-CIRCUITO

LUCIANO DUARTE NASCIMENTO FRANCO

DISSERTAÇÃO SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA MECÂNICA DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA, COMO PARTE DOS REQUISÍTOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM SISTEMAS MECATRÔNICOS.

APROVADA POR:

Prof. Guilherme Caribé de Carvalho, PhD. (ENM-UnB)
(Orientador)

Prof. Sadek Crisóstomo Absi Alfaro, PhD. (ENM-UnB)
(Examinador Interno)

Prof. Américo Scotti, PhD. (UFU)
(Examinador Externo)

BRASÍLIA/DF, 26 DE DEZEMBRO DE 2007.

FICHA CATALOGRÁFICA

FRANCO, LUCIANO DUARTE NASCIMENTO

Sincronização, captura e análise de imagens da poça de soldagem no processo GMAW convencional, no modo de transferência metálica por curto-circuito. [Distrito Federal] 2007.

143p., 297 mm (EMN/FT/UnB, Mestre, Sistemas Mecatrônicos,2007).

Dissertação de Mestrado - Universidade de Brasília.

Faculdade de Tecnologia.

Departamento de Engenharia Mêcanica.

1. Monitoração da poça de soldagem
2. GMAW
3. Sistemas de visão
4. Sincronização em curto-circuito

I. EN/FT/UnB

II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

FRANCO, L. D. N. (2007). Sincronização, captura e análise de imagens da poça de soldagem no processo GMAW convencional, no modo de transferência metálica por curto-circuito. Dissertação de Mestrado em Sistemas Mecatrônicos, Publicação - ENM.DM-17 A/07, Departamento de Engenharia Mecânica, Universidade de Brasília, Brasília, DF, 143p.

CESSÃO DE DIREITOS

NOME DO AUTOR: Luciano Duarte Nascimento Franco.

TÍTULO DA DISSERTAÇÃO DE MESTRADO: Sincronização, captura e análise de imagens da poça de soldagem no processo GMAW convencional, no modo de transferência metálica por curto-circuito.

GRAU: MESTRE ANO: 2007

Luciano Duarte Nascimento Franco
AOS 05 BL B apto 611; CEP:70.660-052 Brasília - DF - Brasil.

DEDICATÓRIA

Este trabalho é dedicado a Deus,
à querida Vilminha, minha mãe, ao meu Pai, às outras mães de coração, à Carol, aos irmãozinhos e amigos.

AGRADECIMENTOS

Agradeço gentilmente a todos que sempre me incentivaram a realização desse trabalho como os prestativos amigos: Marrocos, Daniel, Fernand, Eber , Jones, Frederico, Ginani, Baiano, Anderson, Luis Toledo,
Gerard, Magno, Andre, Vailton, Catalão, Bruno, Munhoz...

E não esquecendo dos professores: Guilherme Caribé, Carlos Llanos , José Maurício, Sadek, Brito, Flamínio...

RESUMO

SINCRONIZAÇÃO, CAPTURA E ANÁLISE DE IMAGENS DA POÇA DE SOLDAGEM NO PROCESSO GMAW CONVENCIONAL, NO MODO DE TRANSFERÊNCIA METÁLICA POR CURTO-CIRCUITO

Autor: Luciano Duarte Nascimento Franco
Orientador: Guilherme Caribé de Carvalho
Programa de Pós-graduação em Sistemas Mecatrônicos
Brasília, Dezembro de 2007

Esse trabalho apresenta o desenvolvimento de um sistema de visão computacional em tempo real que visa a capturar e medir imagens do processo de soldagem **GMAW** em modo curto-circuito. A captura dessas imagens deve estar sincronizada à detecção de um curto-circuito visando a obter imagens com pouco ou nenhum brilho do arco para que um algoritmo de processamento de imagens possa aferir a posição do arame de soldagem e da poça fundida.

Durante o desenvolvimento desse sistema foram estudadas as principais tecnologias envolvidas como: o processo de soldagem GMAW, o porquê de se escolher a soldagem em modo curto-circuito, a analise e monitoração dos sinais de soldagem como a tensão e a corrente, a determinação do melhor ponto para captura de imagens.

Em seguida foi desenvolvida toda a infra-estrutura para criação e experimentação do sistema: a montagem de um sistema computacional industrial de captura de imagens, uma mesa linear para o deslocamento do corpo de prova, a escolha do mais adequado sistema de detecção e sincronização de imagens e o desenvolvimento e validação de uma série de *softwares* como o de detecção de curto-circuitos e o de captura e medição de imagens.

Os resultados desse sistema atingiram os objetivos propostos, pois foi possível obter, imagens de boa qualidade, sem saturação, sincronizadas com a detecção do curto-circuito, permitindo então a medição da poça o processamento robusto de imagens da poça de solda.

ABSTRACT

SYNCHRONIZATION, CAPTURE AND ANALYSIS OF WELD POOL IMAGE IN DIP MODE OF CONVENTIONAL METAL TRANSFER GMAW PROCESSES

Author: Luciano Duarte Nascimento Franco
Supervisor: Guilherme Caribé de Carvalho
Programa de Pós-graduação em Sistemas Mecatrônicos
Brasília, December of 2007

This work presents the development of a real time computer vision system that seeks to capture and to analyse images of a **GMAW** arc welding process in short circuit mode. The capture of those images should be synchronized with a short circuit the detection to obtain images with little or no bright so a image processing algorithm can check the position of the welding wire and molten pool.

During the development of this system were studied the main technologies involved as: the GMAW, weld process, the reason for choosing the shor-circuit welding mode, the analyzes and the monotoring of the welding signs as the voltage and the current, the determination of the best point to capture images.

Meanwhile a whole infrastructure were developed to crate and experiment the system: the assembly of a image capture industrial computer, a linear table to displace the proof body, the choice of a synchronization and detection image system, the development and validation of a series of softwares as the short-circuit detection and the image capture and mesure algorithm.

The results of that system were satisfactory, because it was possible to obtain, in a very efficient way, good quality images, without bright, synchronized with the detection of the short circuit, allowing to measure the molten pool.

SUMÁRIO

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE SÍMBOLOS, NOMENCLATURA E ABREVIAÇÕES

GMAW: *Gas metal arc welding.*

CMOS: *Complementary metal oxyde semiconductor.*

CCD: *Charged coupled device.*

MIG: *Metal inert gas.*

MAG: *Metal active gas.*

GRACO: Grupo de automação e controle.

FPGA: *Field programmable gate arrays.*

USB: *Universal serial bus.*

TIG: *Tungsten inert gas.*

AD: Analógico digital.

RGB: *Red green glue.*

ATA: *Advanced technology attachment.*

PCI: *Peripheral component interconnect.*

API: *Application programming interface.*

RAM: *Random acess memory.*

ROI: *Regions of interest.*

EPI: Equipamento de proteção individual.

TCP: *Tool center point.*

PICMG: *Industrial computer manufacturers group.*

IDE: *Integrated development environment.*

SCSI: *Small computer system interfac*e.

DDR: *Double-data-rate synchronous dynamic random access memory*

MTBF: *Mean time between faliure.*

UnB: Universidade de Brasília.

ASCII: *American standard code for information interchange.*

MFC: *Microsoft foundation classes.*

SDK: *Software development kit.*

VI: *Vitrual instrument.*

MR: Taxa de consumo do eletrodo.

α, β: Constante de aquecimento do tipo eletrodo.

l: *Stick-out* Comprimento entre o eletrodo e o material base.

a: Secção da área transversal do eletrodo.

W_i: Janela de tamanho definido N com as amostras da tensão do arco.

W_{medio}: Tensão média das amostras da tensão do arco.

W_{Minf}: Janela de tamanho definido N com valores menores que a média das amostras da tensão do arco.

W_{Msup}: Janela de tamanho definido N com valores maiores que a média das amostras da tensão do arco.

W_{Mbax}: Tensão média dos valores abaixo da média.

W_{Malt}: Tensão média dos valores acima da média.

Gb/s: Giga *bits* por segundo.

GB/s: Giga *bytes* por segundo.

Mb/s: Mega *bits* por segundo.

MB/s: Mega *bytes* por segundo.

W_{Malt}: Tensão média dos valores acima da média.

$standoff$: Distância entre a extremidade do bocal de gás e o metal de base

1 INTRODUÇÃO

O processo de soldagem **GMAW** convencional, em modo de transferência por curto-circuito, é muito utilizado na indústria devido a características tais como baixo aporte térmico e pequena penetração. Essas características permitem que esse processo de soldagem seja aplicado em uma variedade de aplicações, que incluem desde soldagem de chapas finas a soldagem posicional.

Um problema do processo de soldagem é necessidade de utilização intensiva de mão de obra especializada, a qual tem se tornado escassa e cara. Para contornar esse problema, as indústrias têm substituído o trabalho manual por sistemas de soldagem automatizada, utilizando desde sistemas simples de automação de deslocamento, no caso de soldagem de grandes estruturas tais como cascos de navios, até sistemas complexos de soldagem robotizada, como se observa na indústria automobilística.

Sistemas de soldagem robotizada geralmente são rápidos e robustos e possuem repetibilidade elevada. Entretanto, não conseguem substituir com facilidade o ser humano na tarefa de controlar o processo de soldagem. Um operador de soldagem qualificado utiliza seus sentidos de visão e de audição para detectar desvios de qualidade do cordão depositado. Ao detectar esses desvios, ele atua sobre o processo, com base em sua experiência, seja corrigindo parâmetros de soldagem ou a posição da tocha em relação à junta ou ambos em conjunto, de modo a manter a qualidade requerida. Portanto, trata-se de um sistema de controle em malha fechada de alta complexidade, ainda não replicado em sistemas de soldagem robotizada comercialmente utilizados.

As formas de controle atualmente aplicados em sistemas de soldagem robotizada podem ser classificadas em controle em malha aberta e em malha fechada. O controle do processo em malha aberta confia que o sistema robótico repita a operação sem erro e que a peça a ser trabalhada apresente consistência dimensional e posicional. Para que isso seja possível, precisam-se definir procedimentos de soldagem adequados às exigências de qualidade do produto, além de se projetar corretamente a peça a ser trabalhada assim como seu sistema de fixação para soldagem. Já o controle em malha fechada permite que o sistema robotizado realize correções de desvios de qualidade em tempo de soldagem, tentando simular a ação de um soldador experiente. Para isso, um

ou mais sensores são utilizados para coletar informações a respeito do processo e um sistema de controle realimentado produz as saídas de controle necessárias à manutenção da qualidade requerida.

Os parâmetros de soldagem mais monitorados, tradicionalmente, são a tensão e a corrente de soldagem. Esses parâmetros são muito utilizados por serem conhecidos e por necessitarem de equipamentos de baixo custo para seu monitoramento. Outros parâmetros que podem ser monitorados são os espectros luminoso e sonoro do arco elétrico e a geometria da poça de soldagem, por meio da visão computacional. Esses parâmetros atualmente não são comercialmente utilizados por apresentarem baixa robustez e alto custo.

O rápido desenvolvimento da eletrônica com grande redução de custos propiciou o aparecimento de poderosos sistemas computacionais que possibilitaram, ao longo dos anos 1980 e 1990, o aparecimento de diversos sistemas de monitoramento e controle de processos de soldagem em tempo real. Os principais parâmetros monitorados são tensão e corrente de soldagem, velocidade de alimentação do arame, posicionamento da tocha. Alguns sistemas experimentais utilizando visão computacional foram propostos, porém não foram implementados comercialmente.

Até a década de 1990, havia estudos e análises sobre o uso de sistemas de visão computacional na área industrial para fins de inspeção pós-soldagem
(Andersen, K., Joel e Cook 1992). Nesses estudos, concluiu-se a impossibilidade de aplicá-los em tempo real com os computadores e sistemas de vídeo da época. O desenvolvimento e a redução de custos das tecnologias de captura imagens **CMOS** e **CCD** permitiram que surgissem os primeiros sistemas de visão computacional em tempo real comerciais aplicados a soldagem.

A vantagem em utilizar um sistema de visão juntamente com o monitoramento de parâmetros elétricos do processo de soldagem, além de não intrusivo, reduz de o tempo de exposição do operador a ambientes insalubres. Propicia também, a possibilidade de se ter um registro contínuo e ininterrupto da produção. Entretanto, um dos problemas em se utilizar sistemas de visão computacional em ambientes de soldagem é que a intensidade luminosa proveniente do arco elétrico (coluna de plasma), que, de modo geral, satura grande parte das imagem capturadas (Koike 1999). Isso dificulta o processamento das imagens, acarretando falta de robustez na sua análise e, conseqüentemente, produzindo um ruído nos sinais de controle resultantes, o que impossibilita seu uso

comercial.

Para contornar o brilho do arco elétrico e conseqüentemente melhorar a qualidade das imagens obtidas, esse trabalho apresenta o estudo e a aplicação de técnicas que reduzem ou eliminam o brilho do arco elétrico, por meio de um sistema de visão computacional integrado a sistemas de monitoramento dos parâmetros elétricos do processo de soldagem **GMAW**. Este trabalho também apresenta propostas de melhoria nos algoritmos de processamentos de dados apresentados em (Koike 1999) e propostas de novos algoritmos de medição da geometria da poça, com o intuito de melhorar a eficiência computacional do mesmo.

1.1 OBJETIVO

O objetivo geral deste trabalho é desenvolver um sistema de visão computacional destinado ao monitoração da geometria da poça de fusão durante o processo de soldagem **GMAW** convencional, no modo de transferência por curto-circuito. Para atingir este objetivo, os seguintes objetivos específicos deverão ser alcançados:

- Determinar características necessárias ao sistema de visão para a visualização direta da poça;
- Estudar características do processo **GMAW** e determinar o melhor momento para se obter a imagem da poça de fusão sem saturação da imagem proveniente do arco elétrico;
- Integrar um sistema de medição de parâmetros de soldagem (tensão e corrente) e desenvolver técnicas de detecção do melhor momento para captura da imagem da poça;
- Sincronizar a captura das imagens com o sistema de medição de parâmetros para obter imagens não saturadas da poça de soldagem;
- Desenvolver técnicas de processamento de imagem para analise alguns parâmetros geométricos da poça.

1.2 ORGANIZAÇÃO DO TEXTO

O capítulo 1, a introdução, apresenta um texto introdutório, a motivação para execução desse trabalho, além disso apresenta os objetivos a serem atingidos.

O capítulo 2, apresenta a revisão bibliográfica, onde são abordados: os temas básicos de um processo de soldagem **GMAW**, a monitoração tradicional do processo de soldagem, além disso são apresentados temas relacionados a visão computacional como: a captura e formação de imagens; eficiência e sincronização na captura de imagens; análise de uma imagem da poça de soldagem.

O capítulo 3, a montagem experimental do projeto, onde são apresentados: justificativas do uso de um computador industrial; a montagem da infra-estrutura dos experimentos; a realização dos experimentos básicos: os problemas, soluções apontadas, desenvolvimento de software de captura de imagens; a detecção e sincronização da presença de um curto-circuito; Além disso, são apresentados os resultados dos experimentos avançados os resultados.

O capítulo 5, a conclusão, apresenta o desenvolvimento, os problemas, as soluções e resultados desse trabalho, além disso, propõe alguns trabalhos futuros, a partir do desenvolvimento desse trabalho.

2 REVISÃO BIBLIOGRÁFICA

2.1 PROCESSO DE SOLDAGEM GMAW

Processos de soldagem **GMAW** são baseados em técnicas de arco elétrico, ou seja uma fonte de energia elétrica pode gerar um arco entre duas peças metálicas (o eletrodo e o metal base) quando existem condições propícias para formação do mesmo(temperatura elevada e uma atmosfera ionizada). Durante a existência do arco, grande quantidade de energia é liberada em forma calor. Este permite a fusão do metal base e da ponta do eletrodo, que é alimentado de forma constante, e protegido por uma atmosfera gasosa (Norrish 1992).

Tomando uma exemplo prático, um sistema de soldagem **GMAW** típico mostrado na Figura 2.1 é composto por uma fonte de soldagem (A), uma bobina de arame de soldagem (B) e o respectivo tracionador (C), um cilindro de gás de proteção (D) e uma tocha de soldagem (E). Esta é composta por um bocal condutor que acomoda três conexões uma para o gás de proteção, uma para a fonte de corrente e uma para o arame de soldagem A Figura 2.2 exibe um clássico esquemático do que ocorre durante uma soldagem **GMAW**.

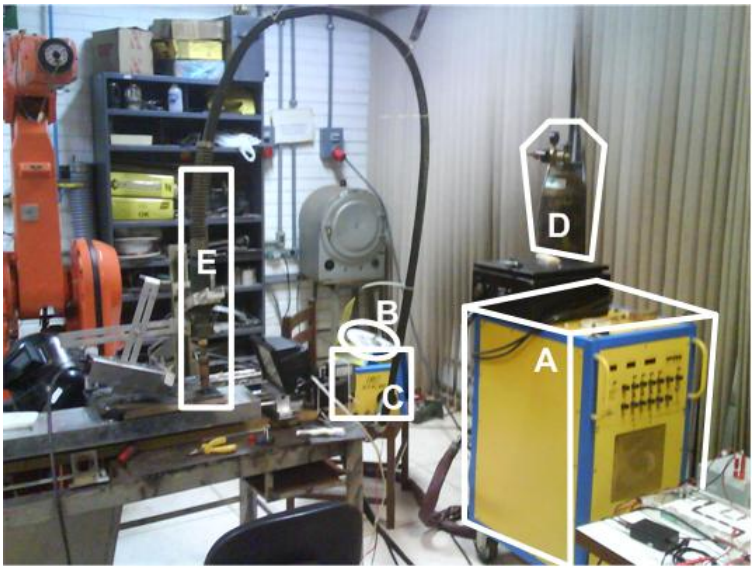

Figura 2.1: Exemplo típico de um sistema de soldagem **GMAW** robótico

O processo do soldagem **GMAW** é o preferido e mais difundido na área industrial, pois,

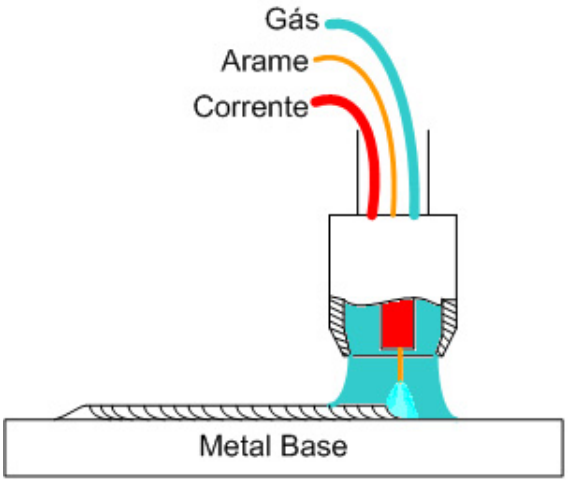

Figura 2.2: Tocha de soldagem **GMAW**

permite a fácil robotização do sistema, principalmente, devido ao arame ser contínuo. Isto favorece uma alto ciclo de trabalho (Marques, P., Modenesi e Bracarense 2005).

Por isso dependendo do material, pode-se usar duas vertentes do processo **GMAW**: **MIG**(*Metal Inert Gas*) onde os gases de proteção são ricos em gases inertes, como o argônio, indicado para quase todos os processos. **MAG** (*Metal Active Gas*) onde os gases de proteção são uma mistura entre gases inertes e ativos como o CO_2 indicados principalmente para materiais ferrosos (Marques, P., Modenesi e Bracarense 2005).

No processo de soldagem **GMAW**, além de se alterar o gás de proteção, deve-se modificar a tensão de soldagem e a corrente ou a velocidade alimentação do arame. Esses fatores afetam diretamente o comportamento do arco e, conseqüentemente, a forma como o arame é transferido sobre o material base e como o cordão de solda é formado. Esses modos são conhecidos como modos de transferência metálica
(Marques, P., Modenesi e Bracarense 2005) e (Norrish 1992).

2.1.1 Modos de transferência metálica

Os modos de transferência metálica são uma forma genérica, aproximada, de classificar o tipo de cordão de solda gerado por cada combinação de parâmetros além de indicar a estabilidade de cada processo. Esse modos foram classificados somente por meio da análise de dois parâmetros de soldagem a tensão e corrente.

Existem três modos de transferência representados na Figura 2.3: curto-circuito que possui predominantemente tensões mais baixas; globular que possui predominante-

mente tensões elevadas e correntes baixas; spray possui tensão e corrente elevadas. Note que existe uma zona de transição entre os modos que permite em um mesmo processo de soldagem obter parâmetros de curto-circuito, globular ou spray no mesmo cordão de solda.

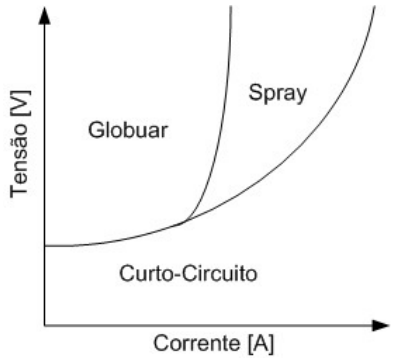

Figura 2.3: Esquemático aproximado do modos de transferência metálica obtidos, por meio de experimentação sucessiva

Por isso é recomendado que, para montagem de um sistema de soldagem automático, deve-se fazer o levantamento desses 3 modos antes do uso a fim de obter com certeza os modos desejados.

2.1.2 Formação do cordão de soldagem

Os três modos de transferência metálica formam cordões de solda diferentes assim como arcos de soldagem diferentes, o primeiro deles, o curto-circuito.

A Figura 2.4 descreve um ciclo de soldagem ocorrido em experimento realizado nos laboratórios do **GRACO** (Grupo de automação e controle), esse experimento pode ser interpretado pela equação de consumo do eletrodo Eq 2.1, onde α e β são as constantes de aquecimento do tipo eletrodo, L é o comprimento energizado do eletrodo (*Stick-out*)e a é a secção da área transversal do eletrodo (Norrish 1992).

$$MR = \alpha I + \frac{\beta L}{a} I^2 \tag{2.1}$$

O processo de soldagem incia em (A) (Figura 2.4), o arco ja foi estabelecido a tensão do sistema é alta, e coexistem em equilíbrio taxas de consumo e alimentação de eletrodo.

Com decorrer do tempo, o eletrodo se aproxima do metal base, principalmente devido a taxa de alimentação ser maior que a de consumo.

Quando o eletrodo toca no material base o arco se estingue tensão cai e a corrente aumenta com uma taxa de crescimento limitada pela indutância do circuito de soldagem (C e D).

Nesse momento eleva-se a formação de material fundido e o equilíbrio entre taxa de alimentação e consumo é invertido (E). Nesse período forças magnéticas e a tensão superficial formam um gargalo.

Devido a corrente estar elevada o gargalo é vaporizado e ocorre uma explosão, que dissipa energia e restabelece o arco, reiniciando o processo (E, F e G).

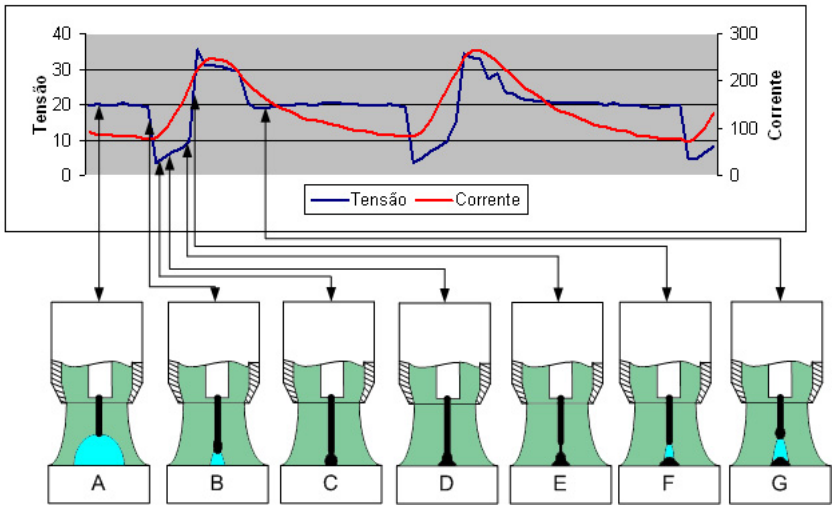

Figura 2.4: Esquemático de modo de transferência por curto-cucurito

A transferência por curto-circuito possui como característica principal a transferência metálica restrita ao período do curto-circuito, além disso, um fato importante utilizado no trabalho é a interrupção total do arco voltaico. Neste período, é o momento ideal para que uma câmera capture da imagem de uma poça fundida
(Adolfsson, S. et al. 1999).

O segundo e o terceiro modos de transferência são o globular e spray respectivamente. A principal diferença entre estes e o curto-circuito é a presença constante do arco voltaico que dificulta a obtenção de uma imagens sem saturação. Mesmo assim ainda é possível obter a poça de soldagem sem brilho, por meio do uso de técnicas de soldagem pulsada que força a ocorrência do fenômeno da transferência em períodos determinados, porém está fora do escopo desse trabalho.

2.2 MONITORAÇÃO DOS PARAMETROS DE SOLDAGEM

Uma importante parte do processo industrial em que é necessário o controle de sistemas de soldagem é a monitoração dos parâmetros do arco, esses parâmetros fornecem detalhes que permitem calibrar equipamentos e detectar possíveis falhas no processo (Norrish 1992).

Um problema associado ao uso de sistemas de monitoração em soldagem é que geralmente estes sistemas são acoplados diretamente à tocha de soldagem e podem interferir no processo.

O exemplo típico é o voltímetro. Este equipamento é colocado em paralelo com o sistema de soldagem e por isso, dependendo da forma como foi montado e como a máquina de soldagem gera o arco inicial (Processos **TIG** utilizam geralmente alta frequência) o equipamento de medição pode funcionar como uma carga não desprezível ao sistema podendo até ser a carga principal da máquina, podendo causar até acidentes.

Outro problema associado a esses tipos de equipamentos de medição de arco de soldagem é o aparecimento de ruído causado por vários harmônicos gerados no processo pela máquina de soldagem.

Em trabalhos passados (Togawa 2003) e (Granja 2001) foi desenvolvido no **GRACO** um voltímetro Figura 3.3 especialmente desenvolvido para trabalhar com este tipo de problema, além disso utiliza um amplificador isolador para desacoplar os sinais recebidos da máquina de soldagem do elemento de medição. Nesse trabalho foi realizada a melhoria desse sistema a fim de diminuir o ruído de soldagem na saída do isolador (item 3.3.1).

Um outro exemplo típico de sistema não intrusivo é o amperímetro de efeito *Hall* já que ele mede a partir do campo magnético gerado. Assim como o voltímetro esse equipamento é suscetível a ruídos externos.

Para analisar de forma adequada parâmetros de soldagem em tempo real em meios digitais é necessário que existam sistemas que convertam a parâmetro medido, exemplo a tensão de entrada, para um valor digital adequado para processamento, por meio de um conversor analógico digital. Esse pode fazer parte de um computador com um dispositivo especial (placa de aquisição) ou de um dispositivo embarcado (um micro-

controlador).

A digitalização de um sinal analógio deve seguir a clássica regra de *Nyquist* ou seja a taxa de amostragem deve ser o dobro da máxima frequência do sinal amostrado por isso segundo trabalhos como (Zhenguo, S. et al. 2006) e (Carlson, N. et al. 1993) a maior taxa sugerida seria de 10 kHz para monitoração da tensão do arco de soldagem.

Nesse trabalho foram desenvolvidas técnicas monitoração de sinais da tensão e corrente do arco para serem processados utilizando computadores ou sistemas embarcados utilizando microcontroladores.

2.3 ANÁLISE DOS PARAMETROS DE SOLDAGEM

Os parâmetros de tensão e corrente de soldagem nesse trabalho foram analisados a fim de detectar a presença de curto- circuito em soldagem **GMAW**. Em trabalhos como ((Carvalho 1997) e (Adolfsson, S. et al. 1999)) utiliza-se a tensão como referência pois ela responde de forma mais rápida que a corrente devido principalmente à indutância presente nas máquinas de soldagem.

Trabalho de (Carvalho 1997) propõe a utilização de técnica de janelamento com o objetivo de determinar os valores mínimos e máximos da tensão que definem a presença do curto-circuito. Esse conjunto médias torna o processo de análise mais simples pois atenuam as variações bruscas e aleatórias geradas pelo processo de transferência por curto-circuito.

O sistema de médias trabalha com janelas W_i de tamanho definido com N amostras da tensão do arco. A partir dessa janela calcula-se a tensão média W_{medio} (Eq. 2.2), então dividem-se as tensões da janela em maior W_{Minf} e menor W_{Msup} que a tensão média e calculam-se a tensão média baixa W_{Mbax} (Eq 2.3) e a tensão média alta W_{Malt} Equação (2.4).

$$W_{medio} = \frac{\sum_{i=1}^{N} W_i}{N} \tag{2.2}$$

$$W_{Mbax} = \frac{\sum_{i=1}^{N_b} W_{Minf}}{N_b} \tag{2.3}$$

$$W_{Malt} = \frac{\sum_{i=1}^{N_p} W_{Msup}}{N_p} \tag{2.4}$$

Um bom exemplo é a figura 2.5 onde são aplicados os conjuntos de média móveis a uma janela capturada de um experimento **GMAW** por curto-circuito note como é possível definir que valores de tensão abaixo da média baixa são realmente um curto-circuito.

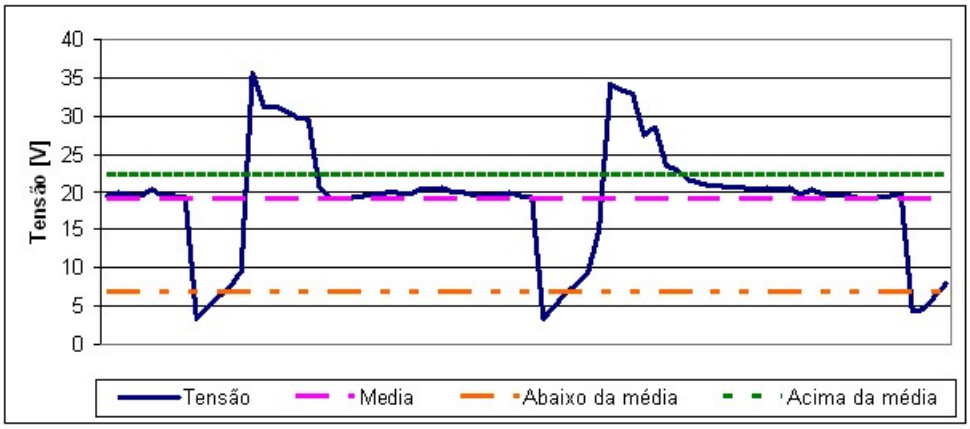

Figura 2.5: Exemplo de valores médios como limiares

2.4 VISÃO COMPUTACIONAL

O funcionamento de um sistema de visão computacional consiste na união de um conjunto de equipamentos a fim de que seja possível a captura, transmissão e processamento digital de uma imagem.

Um bom exemplo é um sistema muito conhecido é o da *webcam* em que existe uma simples câmera **CMOS** um sistema embarcado da própria câmera que captura e pré-processa alguns parâmetros da imagem, como correção da luz, e transmite esses dados serialmente utilizando o barramento **USB** de um computador que exibe esses dados ao usuário.

Sistemas de visão que utilizam *webcam* podem não produzir bons resultados dependendo do fim da aplicação, por exemplo, vigilância residencial: câmeras semelhantes a *webcam* são boas para ambientes internos com média luminosidade porém para ambientes com grande luminosidade elas saturam a imagem (imagem forma um borrão branco) logo limitam sua aplicação.

Em outro exemplo, *webcam* podem ser usadas em ambientes industriais para processos simples de velocidade e precisão baixas como em aplicações como "existe ou não existe

algo em uma caixa". Este tipo de equipamento foi testado em soldagem do tipo **TIG** no artigo (Balfour, C., Smith e Al-Shamma 2006).

Em ambiente de soldagem **GMAW** esse tipo de equipamento não deve ser utilizado devido à falta de robustez e principalmente à baixa velocidade de amostragem. Sistemas de soldagem **GMAW** em modo de curto-circuito possuem um ciclo (tempo entre curtos-circuitos) próximo a 7 ms, em média, podendo chegar a 1 ms (Norrish 1992). Isso significa uma amostragem de 140 fps a 1000 fps. Uma *webcam* típica possui uma taxa de amostragem de 30 fps ou menor, o que a impossibilita de ser utilizada nessa finalidade.

Outros motivos para não se usar uma *webcam* em um processo **GMAW** são: a dificuldade em se adaptar, uma objetiva ou um filtro à mesma, o tipo de sensor de imagem utilizado de qualidade e resolução baixos ou duvidosos, a inexistência de controle de luminosidade mecânico e de sincronização de captura.

Por esses motivos um sistema de visão computacional para soldagem **GMAW** deve possuir uma taxa de amostragem alta, um sensor de imagens de alta qualidade e resolução, um conjunto de objetivas com abertura ajustável, um conjunto de filtros ópticos e um sistema eficiente de sincronização.

2.4.1 Formação de imagens digitais

O funcionamento de uma câmera digital possui o mesmo princípio básico da clássica câmera *pin-hole* (câmera de latinha) Figura 2.6. Ao compara-lá a uma câmera digital, fica evidente que foi inserida uma objetiva para obter e concentrar mais a luz e o anteparo que possuía um filme fotossensível foi substituído por um sensor de imagens fotoelétrico.

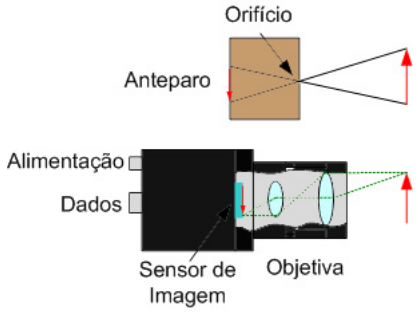

Figura 2.6: Comparação entre câmera digital e *pin-hole*

Um sensor de imagens típico é formado por uma matriz de milhares de foto sensores individuais esses sensores respondem à energia recebida de um fóton e a convertem um sinal elétrico de forma semelhante a um conversor **AD**. Um sensor de imagem responde de forma eficiente a uma certa faixa de comprimentos de onda geralmente a faixa do visível (350 a 700nm), e responde com menor intensidade a comprimentos acima ou abaixo dessa faixa. Isso é caraterizado pelo gráfico da responsividade espectral.

A responsividade espectral é um fator importante no trabalho, pois a poça fundida é melhor vista em comprimentos de onda na faixa do vermelho (650 a 750 nm) para o infravermelho baixo (750 1400 nm) (Koike 1999) e
(Balfour, C., Smith e Al-Shamma 2006) . A figura 2.7 mostra o sensor de imagem da câmera utilizada em nosso trabalho (a esquerda) e o respectivo gráfico de responsividade (à direita) . Note que a câmera possui uma pequena reposta no campo do infravermelho mas possui uma boa resposta ao vermelho.

A inovação em câmeras digitais restringe-se à parte eletrônica já que, as objetivas são legados das câmeras analógicas tradicionais. Atualmente existem dois tipos de sensores de imagens o **CCD** (*charged coupled device*) e os sensores de tecnologia **CMOS** (*complementary metal oxyde semiconductor*) cada uma com suas vantagens.

A tecnologia **CCD** possui melhor qualidade de imagem que a **CMOS**, porém esta é mais rápida e relativamente mais barata. A tecnologia **CMOS** possui um ruído de fundo bem maior que a **CCD**, porém uma vantagem exclusiva da tecnologia **CMOS** é possibilidade de definir individualmente regiões de sensibilidade do sensor de imagens (Litwiller 2001). Essas características definem a câmera **CMOS** como aplicável a monitoração do processo de soldagem.

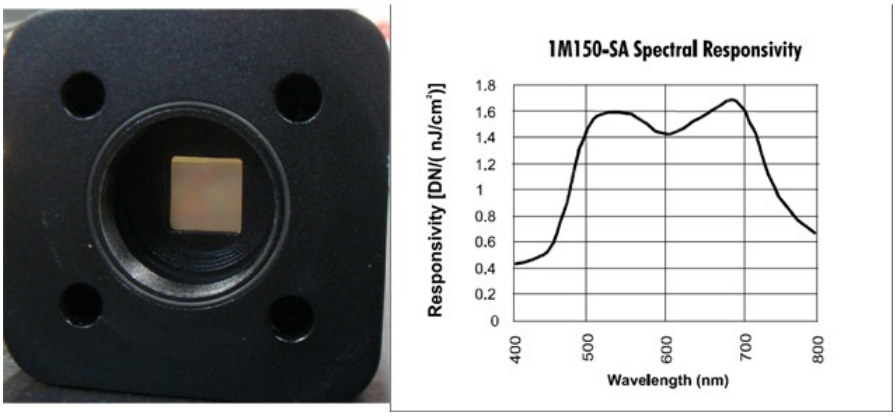

Figura 2.7: Sensor de imagem e a responsividade espectral da câmera utilizada nesse trabalho

Sabe-se que na tecnologia **CMOS** cada fotosensor possui uma posição e uma intensi-

dade luminosa independentes logo um conjunto deles são agrupados para formar uma matriz chamada de sensor de imagens (Litwiller 2001). Esse elemento unitário da matriz é chamado *pixel*, quanto mais *pixels*, melhor a definição de imagem. Um exemplo, a câmera utilizada nesse trabalho, possui 1024 linhas e 1024 colunas ao realizar essa simples conta $\frac{Linha*Coluna}{10^6}$ obtém-se a resolução da câmera em *megapixels*.

Um exemplo de como uma imagem é representada em uma matriz é a Figura 2.8 que contém uma poça de soldagem de um experimento, note como ela é composta na totalidade em (A) e veja o detalhe (B) e como a câmera monocromática de 8 bits representa a quantização os *pixels* em valores que podem variar de 0 a 255 e são representados cores diferentes da escala de cinza.

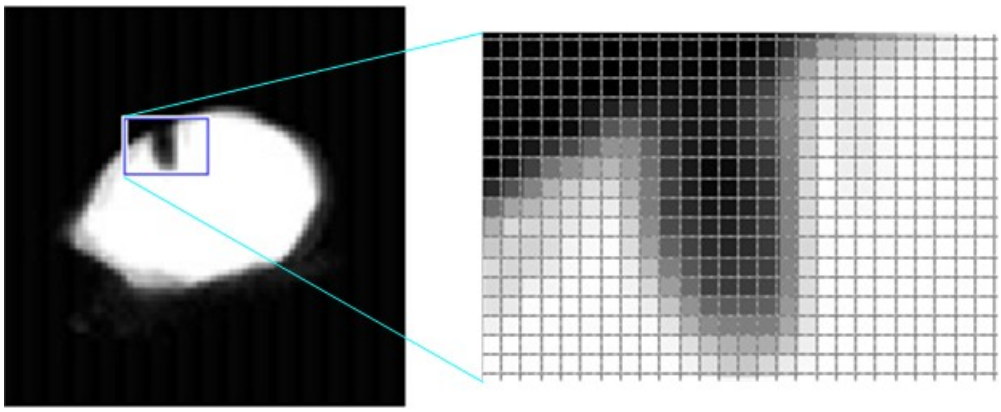

Figura 2.8: Como uma figura é representada em uma matriz de dados

2.4.2 Captura de imagens

Em sistemas de visão computacional quando um *frame* é capturado ele deve ser apresentado em formato digital ao barramento de dados de um computador a taxas de transmissão adequados ao tipo de câmera.

A parte de apresentação de dados é atualmente é um problema crítico na área de captura de video pois sistemas, até pouco tempo, possuiam taxas de transmissão muito baixas em relação a taxa de quadros que uma câmera poderia obter. Um exemplo a câmera utilizada nesse trabalho possui uma taxa de amostragem máxima de 150 fps com resolução de 1 *Megapixel* com 8 bits de tons de cinza por pixel por quadro. Isso implica que em 1 segundo o canal de transmissão e o sistema de armazenamento de imagens necessita obter 150 quadros de 1 *Megabyte* uma banda de 1200 Mb/s. Imagine o mesmo sistema a cores de 8 bits e 3 canais de cor **RGB** - *red green glue* podendo chegar até 3600 Mb/s.

No passado, várias empresas produziram diferentes protocolos e formas de receber, a velocidades adequadas, esses dados porém isso tornou o mercado confuso até várias empresas de grande porte padronizarem em um sistema adequando a forma como os dados e o controle das câmeras seriam realizados. Este se chama **CAMERA LINK** (Camera Link 2000).

Esse protocolo é aceito pela maioria das câmeras e pode atingir taxas teóricas de 1923 Gb/s (Camera Link 2000), atualmente *frame grabbers* que utilizam este protocolo estão na faixa de 4 Gb/s.

Com o desenvolvimento da tecnologia embarcada estão aparecendo as *smart cameras* que tem como intuito embarcar o *frame grabber* e também já processar dados independentemente de um computador a fim de baixar custos, e utilizar tecnologias de transmissão de dados considerados padrões como *serial* ***ATA*** com até 3Gb/s, *Giga-Bit Ethernet* de 1Gb/s, ***USB*** *2.0* de 480 Mb/s ou *Firewire* 400Mb/s todos eles para somente transmitir os resultados já processados.

2.4.3 Eficiência na captura de imagens

Mesmo que um sistema de captura e transmissão de dados seja eficiente na entrega de dados a um microcomputador o sistema computacional que recebe deve ser capaz de receber e processar a vazão de dados que um **frame grabber** possa transmitir. Em sistemas computacionais que trabalham com visão computacional em tempo real tem-se restrições tanto em *hardawre* quanto em *software*.

Em um microcomputador atual comum, geralmente o barramento para instalação de placas é o **PCI** que é limitada a uma taxa de dados a 133 **MB**/s (1064 **Mb**/s), por isso para que um computador seja capaz de receber dados de um *frame grabber* com velocidades máximas de 500 MB/s ele deve possuir um barramento especial.

Para isso deve-se usar um barramento especial chamado **PCI-X** (não confunda com **PCI-Express**) que pode ser encontrado em computadores corporativos e industriais que necessitam receber grande quantidade de dados. As versões conhecidas são o **PCI-X** 64, 133, 266, 533. O mais lento deles, o **PCI-X** 64, é utilizado nesse trabalho com taxas máximas de 533 MB/s (4264 Mb/s) e o mais rápido **PCI-X** 533 pode chegar a 4266 MB/s (34128 Mb/s) (Digi International 2003).

Para que um software possa processar dados de forma eficiente é necessário haver uma

série de técnicas que visam a otimizar formas de processamento, distribuir e reduzir o processamento de quadros.

Uma forma de otimizar o processamento é criar um *software* de baixo nível otimizado. Geralmente, o fabricante do *frame grabber* fornece esse software por meio de **API** (*application programming interface*). As **API´s** são programas pré-compilados geralmente protegidos que são adicionados a projetos de *software* a fim de facilitar e acelerar o desenvolvimento do software oferecendo ao programador somente funções simples de acesso aos dispositivos isso oculta detalhes de implementação de baixo nível que o programador provavelmente desconhece e nem necessita saber.

A **API** fornecida pelo fabricante deve utilizar de forma intensa ponteiros associados a uma imensa quantidade de vetores reservados estaticamente em memória **RAM** (*random access memory*) a fim de armazenar vários quadros de imagens e evitando ao máximo o uso do disco rígido, que possui um tempo de acesso da ordem dos milissegundos.

Outra forma de processamento é distribuir o mesmo com dispositivos dedicados, alguns *frame grabber´s* possuem **FPGA** (*field programmable gate arrays*) que permitem que seja possível implementar algumas partes de código como filtros diretamente em hardware (Dalsa 2003) . Nesse trabalho não foram estudados esses tipos de otimização.

Para otimizar o processamento pode-se reduzir e ou restringir o tamanho do quadro capturado, por exemplo um quadro fornecido por uma câmera possui 1024 linhas e 1024 colunas será que realmente é necessário o uso dessa resolução, ou será necessário utilizar somente um pedaço da imagem o invés do todo? Na maioria das vezes em aplicações de visão computacional deseja-se processar somente uma regiões do quadro.

Para isso exitem tipos de captura que utilizam o **ROI** (*regions of interest*) que definem diretamente na câmera ou no *frame grabber* uma área de captura isso diminui o número de *bytes* por quadro além de poder aumentar a taxa de captura dos quadros.

Veja o exemplo da Figura 2.9 o pequeno retângulo definindo uma área de interesse pode reduzir um quadro de 480 colunas por 320 linhas que ocupa de 151 KB de espaço em disco para quadro muito reduzido, mas eficiente de 94 colunas por 44 linhas que ocupa de 5 KB.

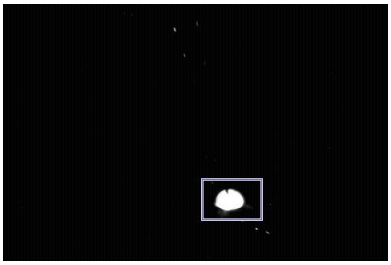

Figura 2.9: Exemplo de uma imagem com o uso de ROI

Uma câmera pode fornecer continuamente imagens, mas será que isso é realmente necessário? Na maioria das vezes, pode-se reduzir o processamento a somente imagens que interessam ao invés de processar um conjunto. Para isso um dispositivo pode disparar a captura, um bom exemplo são os controladores de velocidade eletrônicos que capturam imagens do veículo infrator somente quando um detector de velocidade instalado diretamente no asfalto indica se um veículo acima da velocidade acabou de passar.

2.4.4 Sincronização e captura de imagens

Nesse trabalho foram estudadas várias formas de sincronizar a captura da imagem à detecção de um curto-circuito com intuito de reduzir o número de *frames* e capturar a melhor imagem sem brilho.

Segundo (Adolfsson, S. et al. 1999) o melhor período para captura de imagens ou seja o de menor intensidade de brilho em um processo **GMAW** ocorre quando o arame toca a peça de soldagem. Isso extingue o arco deixando somente o brilho da poça de soldagem fundida.

Por isso foram sugeridas 3 técnicas para detecção do curto-circuito e envio do sinal de sincronização para a câmera ou para o *frame grabber*. Duas utilizando sistemas computacionais e redes *ethernet* e outra totalmente via *hardware*. Essas são descritas e testadas em 3.3.6.

2.4.5 Análise de imagens

Em sistemas de visão computacional, mais que o uso eficiente de captura e processamento, deve-se ter em mente que é necessário utilizar uma série de algoritmos que funcionam como ferramentas que visam a: melhorar, acentuar, atenuar, filtrar e analisar parâmetros ou características da imagem.

A forma mais simples de melhoria de imagem é o uso de filtros ópticos físicos que podem atenuar, impedir ou filtrar determinados comprimentos de ondas que são recebidos por um sensor de imagens de uma câmera. De acordo com
(Balfour, C., Smith e Al-Shamma 2006) e (Koike 1999) o uso de filtros ópticos na faixa do infravermelho reduziriam o brilho do arco de soldagem exibiram a poça de soldagem incandescente.

A melhoria das imagens pode ser realizada por meio de filtros digitais que podem atenuar consideravelmente o ruído de fundo gerado por exemplo pelo sensor de imagem **CMOS**. O filtro mais comum a ser utilizado é o *threshold* que é uma simples forma binária de separação de dados para que, arbitrariamente definido um limiar, tudo abaixo disso seja descartado ou transformado em zero.

Esse valor arbitrário do filtro *threshold* pode ser definido por técnicas como analise de frequências de um histogramas ou utilizando janela de médias aplicadas à imagem em estudo.

A partir de uma imagem de poça de soldagem (Figura 2.10) é possível retirar uma série de informações quantitativas como: área, diâmetro, altura, posição do arame soldagem, posição em relação a um centro de referência. Existem algoritmos que conseguem medir parâmetros quantitativos de poça utilizando principalmente técnicas de cálculo numérico (Koike 1999).

Figura 2.10: Exemplo de uma poça de solda

Existem também características qualitativas da poça como: formato de poça, excesso de respingos. Para medir parâmetros qualitativos não é tão simples. Veja o exemplo da Figura 2.11, onde o formato da poça de soldagem pode ser obtido por meio da comparação por correlação entre uma imagem padrão (B) e a imagem obtida (A). Uma outra técnica é a de utilizar redes neurais do tipo *perceptron* para classificar a poça (Zhenguo, S. et al. 2006). Nesse trabalho não foram implementadas técnicas para aferir as características qualitativas da poça de soldagem.

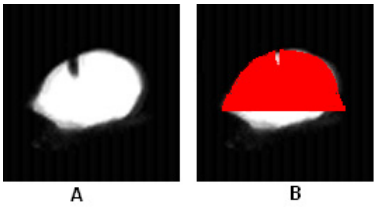

Figura 2.11: Exemplo de possível correlação entre a poça e um padrão arbitrário

De forma quantitativa, para medir a poça de soldagem pode-se aproveitar as características da imagem capturada Note a Figura 2.10 em que fica evidente que a imagem possui apenas a poça de soldagem isso facilita a forma como medir esta imagem, por meio do uso de somas das colunas da imagem vertical e horizontal propostas em artigo por (Koike 1999).

A ideia de utilizar somas é transformar uma imagem bidimensional em um gráfico para então utilizar técnicas de cálculo numérico, como gradientes, a fim de encontrar pontos de inflexão, geralmente relacionados ao aparecimento do início e do fim da borda da poça de soldagem, podendo então definir a diâmetro e altura da poça veja a Figura 2.12.

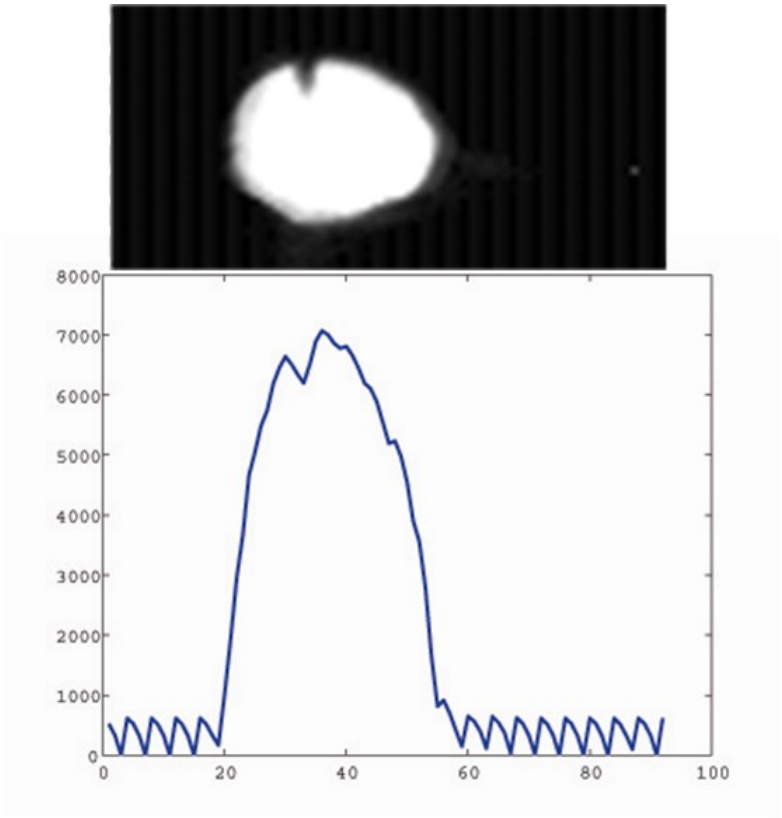

Figura 2.12: Somatório das linhas verticais podem fornecer parâmetros da poça como as bordas

Note na Figura 2.12 que de forma visual é possível inferir a partir do gráfico de somatório das linhas verticais em azul o início e o fim da borda da poça de soldagem e também é possível determinar a posição do arame de soldagem. Note além desses parâmetros que é possível ver o ruído de fundo gerado pelo sensor **CMOS** previsto em 2.4.1. Isso indica a necessidade de aplicação de um filtro de *threshold* para retirada de valor de pontos muito baixos.

2.5 USO DE COMPUTADOR INDUSTRIAL

Ambientes de soldagem são considerados insalubres para seres humanos, por isso é imposto o uso obrigatório de **EPI** (equipamento de proteção individual) como máscaras, luvas, macacões tudo isso visando a segurança do funcionário.

Sistemas computacionais domésticos ou mesmo corporativos são destinados para o uso em ambientes controlados como temperatura e tensão elétrica, porém no chão de fábrica não é possível haver este tipo de ambiente. Um sistema de visão computacional doméstico baseado em *webcam* não resistiria a vibrações excessivas, temperaturas elevadas, picos de tensão elétrica, ruídos na rede elétrica, presença de partículas sólidas vindas de respingos, presença de gases oxidantes em fumos de soldagem, presença de água e risco de acidentes mecânicos como quedas e colisões acidentais.

Esses problemas podem afetar facilmente um sistema de monitoração para soldagem baseados em computadores domésticos. Por isso a necessidade de utilização de uma computador industrial . A grande vantagem do uso desses sistemas é que estes possuem filtros de poeira, placas, gabinetes, fontes, aterramento e espaço otimizados proporcionando mais robustez e desempenho. Nesse trabalho foi iniciado o uso desse tipo de equipamento no **GRACO** e montagem do mesmo é descrito no anexo B.

3 METODOLOGIA E MONTAGEM EXPERIMENTAL

Para o desenvolvimento deste trabalho propôs-se uma metologia baseada em duas linhas de ação: uma consistiu em estudar formas eficientes para que uma câmera de alta velocidade pudesse capturar somente imagens de qualidade da poça fundida de um processo de soldagem **GMAW** em modo de transferência metálica por curto-circuito.A outra consistiu em desenvolver um *software* eficiente que pudesse capturar e analisar as imagens geradas pela câmera de alta velocidade.

Um obstáculo na utilização de câmeras para inspecionar um processo de soldagem é o brilho excessivo, proveniente do arco elétrico, que, em processos com transferência por curto-circuito, é variável no tempo. A intensidade luminosa excessiva limita a qualidade das imagens capturadas, pois o arco elétrico satura as imagens que as transformam em um grande borrão branco que impede a análise da imagem da poça fundida.

Porém há alguns momentos em que o arco de soldagem é extinto. Esses momentos são restritos a uma queda abrupta da tensão do arco, o curto-circuito. Nesse período é o momento ideal para se obter uma imagem, por isso a primeira linha de estudo é encontrar formas eficientes de sincronizar a captura da imagem da poça ao período de menor intensidade luminosa.

Supondo que haja uma sincronização perfeita entre a queda de tensão e a captura de imagens, deve-se implementar um software que permita a captura da imagem digitalizada de forma eficiente, com o mínimo de atraso entre a sincronização e a apresentação da imagem digitalizada, para armazenamento em uma memória rápida, para então, implementar digitalmente algumas algoritmos de medição da poça e então, formar de um registro em tempo real caso outra aplicação que necessite desses dados.

Por isso a segunda linha de estudo concentrou-se em desenvolver e buscar softwares que usassem de forma intensiva o processamento disponível do *hardware* oferecido na máquina, a fim de atingir uma das metas do trabalho: analisar parâmetros da poça de soldagem em tempo de soldagem.

Para atingir os objetivos propostos foi necessário estudar, criar e analisar uma infra-

estrutura física e lógica a qual incluiu:

- A especificação e o desenvolvimento de sistemas mecânicos: suportes para posicionamento e fixação da câmera e da tocha de soldagem **MIG/MAG**, um sistema de posicionamento e movimentação ajustável para peça de soldagem, um sistema de acoplamento e fixação de filtros ópticos, um sistema de proteção do conjunto câmera e filtro;
- A integração de sistemas de medição de parâmetros do processo: um sistema de medição de tensão e de corrente de soldagem, um sistema de geração de pulsos de sincronização (*trigger*) para disparar a captura de imagens a partir da análise dos sinais de tensão;
- Desenvolvimento de software para: detecção de curto-circuito e geração de pulsos de sincronização, captura imagens e medição de parâmetros geométricos da poça

A montagem da infra-estrutura consistiu resumidamente em quatro itens: montagem de um computador industrial, montagem de uma mesa linear digitalmente controlada, construção de sistema de controle remoto de uma máquina de soldagem, definição de um suporte ajustável para câmera.

3.1 MONTAGEM DE UM COMPUTADOR INDUSTRIAL

O sistema de visão computacional foi montado em um computador industrial com perspectivas de em um futuro próximo instalar nesse mesmo computador uma série de outros sensores já disponíveis na **UnB** como o infravermelho, espectrômetro, sensores de tensão e corrente. Todos esses integrados em um poderoso e compacto sistema de monitoração de soldagem.

Devido a complexidade em montagem e configuração desse sistema, foi criado um manual detalhado sobre a montagem e discussão sobre o uso desse tipo de equipamento no anexo B.

3.2 MONTAGEM DE UMA MESA LINEAR

Para realizar os experimentos que envolvam a captura de imagens da poça de soldagem **GMAW** é necessário manter fixo a distância entre a câmera e o ponto de toque do

arame de soldagem ou seja a câmera deve estar sempre apontada e centrada para um ponto próximo a extremidade da tocha de soldagem.

A metodologia utilizada por (Koike 1999) foi acoplar a câmera diretamente em um braço robótico mantendo assim a mesma distância porém um problema relatado nesse tipo de configuração é o aparecimento de uma vibração causada pelo movimento do braço robótico que afetou a aquisição de imagens.

Visando evitar essa vibração foi proposta uma nova metodologia onde a tocha de soldagem e a câmera se manteriam fixas e somente a peça a ser soldada estaria em movimento. Para isso foi montada uma mesa linear com um eixo de livre acoplado a um motor de passo.

A mesa montada é exibida na Figura 3.1. Note que existe uma placa de cobre para facilitar a distribuição rápida do calor e permitir a instalação de um retorno para a corrente de soldagem. Para evitar danos à mesa linear, seja por fuga de corrente ou por aquecimento excessivo, foi instalado sob a placa de cobre uma placa de 5 mm de baquelite. Isso possibilita o isolamento elétrico e térmico.

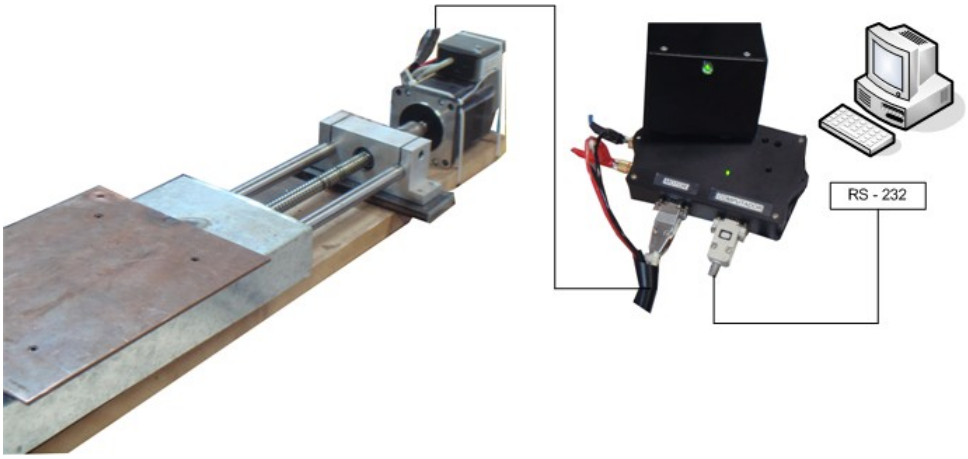

Figura 3.1: Diagrama de montagem da mesa linear

A mesa linear foi encomendada e produzida sob medida essa mesa suporta com segurança uma carga de até 15 Kg. O motor de passo utilizado nessa mesa é do fabricante **BERGER LAHR** modelo **ICIA IDS91** capaz de ser controlada diretamente por meio de um trem de pulsos podendo ser configurada de 200 pulsos por volta até 10000 com um torque de 2 Nm. O microcontrolador utilizado para controlar o motor de passo é do fabricante **MICROCHIP** modelo PIC16C63A.

Para controlar de forma eficiente, primeiro foram definidas as velocidades máximas e mínimas para soldagens do tipo **GMAW** na mesa. Por isso definiram-se velocidades de soldagem de 1 mm/s até 20 mm/s. A mesa linear é capaz de deslocar-se em duas direções, na ordem dos milímetro, em velocidade constante.

Para controlar o motor foi desenvolvida uma placa de controle que possui: (A) um microcontrolador com um *clock* de 4 MHz; (B) três conectores: um para alimentar a placa de controle e o motor de passo com uma tensão de 24 V , um para controlar o sentido e velocidade do motor de passo e outro para receber indicações de falhas do motor; (C) dois Botões para posicionar manualmente a mesa linear; (D) um botão de *reset* para parada de emergência.

A placa de controle da mesa linear foi desenvolvida para ser utilizada em conjunto com um computador, pois recebe comandos como sentido, velocidade e tempo de execução ou distância via porta serial a uma taxa de 4800 bps e responde confirmando a recepção desses comandos e aguarda a ordem de execução.

A Figura 3.2 esboça o protocolo de comunicação entre a placa de controle e um computador que deseja realizar dois tipos de controle: um realizar o deslocamento da mesa a uma velocidade de 4,5 mm/s por 5 segundos em um determinado sentido (1), e outro é deslocar a mesa em a velocidade de 5,6 mm/s por uma distância de 10 mm no sentido oposto ao comando anterior (2). O computador envia os dados em A recebe uma confirmação desses dados enviados pelo microcontrolador em (B) e o computador pode enviar o comando para executar (EXE) ou para cancelar o experimento (CLR) (C). Note que o símbolo $< CR >$ é a representação do valor caractere *enter* (representado na tabela **ASCII** por 13).

A vantagem de se utilizar a porta serial é a possibilidade de usar programas desenvolvidos em *C*, *C++*, *MATLAB*, *Labview*, ou mesmo outro microcontrolador para controlar a mesa com uma precisão adequada ao processo de soldagem **GMAW**.

3.3 MONTAGEM E METODOLOGIA PARA REALIZAÇÃO DO EXPERIMENTO BÁSICO

O objetivo desse trabalho é de sincronizar, capturar e analisar a imagem da poça de soldagem utilizando sistemas de visão computacional. Devido aos vários fatores variáveis como a determinação de curto-circuito, meio de sincronização, tempo de ex-

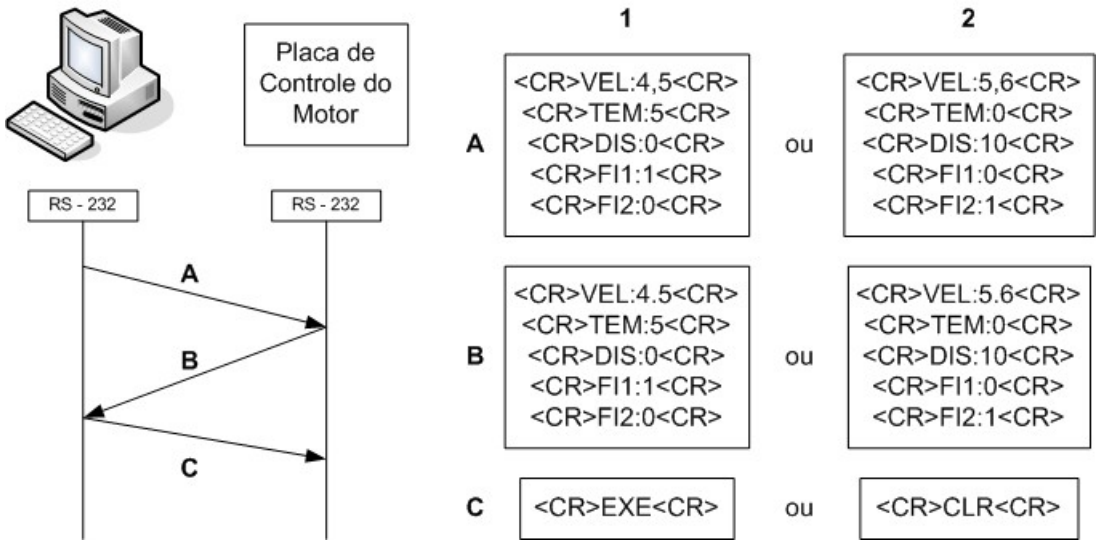

Figura 3.2: Exemplo do funcionamento do protocolo de controle do motor de passo

posição, posicionamento da câmera, uso de filtros ópticos, foi sugerida a divisão em vários pequenos experimentos a fim de desvendar o funcionamento individual de cada subsistema e ao final juntar todos estes em um só grande sistema de de monitoração.

3.3.1 Sistema de medição da tensão do arco de soldagem

Para medir a tensão do processo de soldagem **GMAW** foi proposto utilizar um conjunto de placas de aquisição desenvolvida no **GRACO** e apresentada no item 2.2 desenvolvida pelo nos trabalhos de (Togawa 2003)e (Granja 2001).

A primeira placa, Figura 3.3, é composta por um divisor de tensão na escala 10 para 1, além disso, possui um filtro passa baixa de 2 ordem a fim de reduzir a alta frequência produzida por algumas máquinas de soldagem para abrir o arco no processo **TIG**. A segunda placa é um sistema de isolamento de sinais, cuja função é isolar o circuito de soldagem do circuito de medição, por meio de um amplificador isolador (**ISO122**) e circuitos acessórios, Figuras 3.4, 3.5.

Remontando e testando o equipamento detectou-se uma série de componentes queimados como os conversores DC-DC. Além de um ruido excessivo gerado por componentes já conhecidos pela fama de ruidosos como **LM741**, e a falta de um plano de terra em todas as placas.

Por esses motivos optou-se por simplesmente reconstruir as placas de aquisição seguindo o mesmo circuito das placas antecessoras mas com melhorias na implementação como

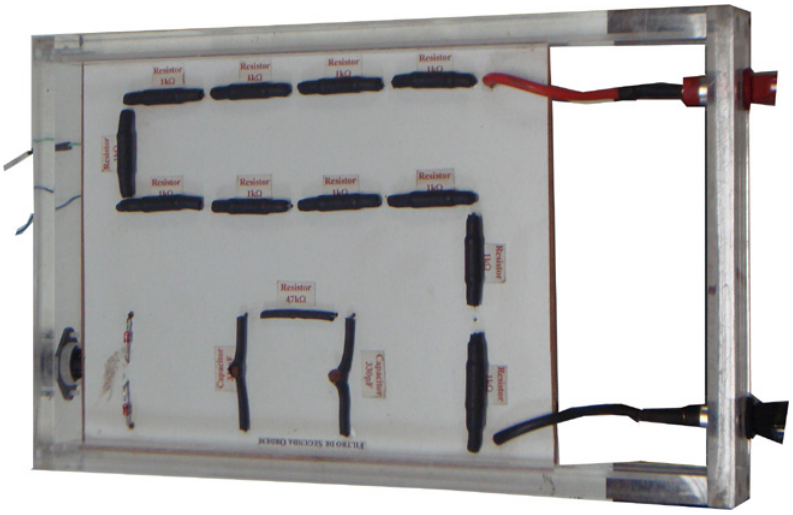

Figura 3.3: Redutor de Tensão 10 para 1

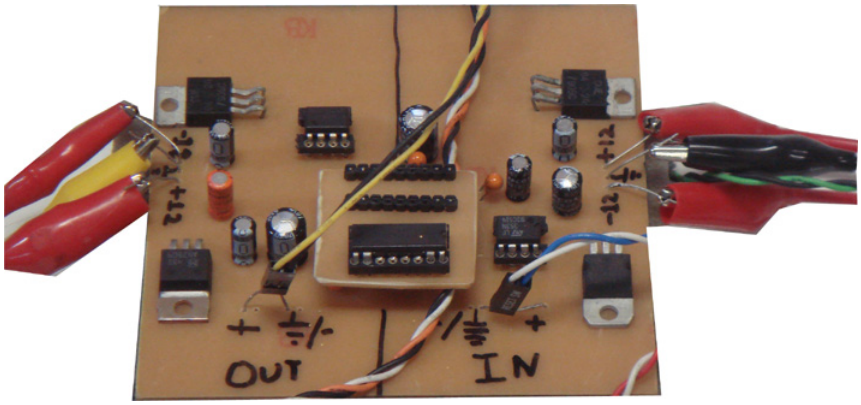

Figura 3.4: Placa de condicionamento de sinal de tensão do arco de soldagem

o uso de planos de terra e aplicação de capacitores de tântalo em todas as entradas dos isoladores tudo com o intuito de reduzir o ruído. A nova placa reduziu consideravelmente o ruido observado anteriormente.

Um problema ao uso dos conversores DC-DC é o alto custo, a necessidade de utilizar reguladores de tensão na saída, a falta de robustez a curto-circuitos acidentais, que geralmente causam a queima do componente que é baseado em transformadores. Por isso foram retirados os conversores DC-DC nessa nova placa de condicionamento.

Os conversores DC-DC foram substituídos por por duas fontes ATX de computadores domésticos adaptadas para gerar uma tensão de +11 e -10 V Figura 3.6. Apesar do tamanho a grande vantagem é que são baratas e facilmente encontradas em sucatas geralmente bastando trocar os fusíveis em caso de curto. Um detalhe dessa configuração é evitar que a carcaça das duas fontes se toquem unindo assim os terras das duas fontes

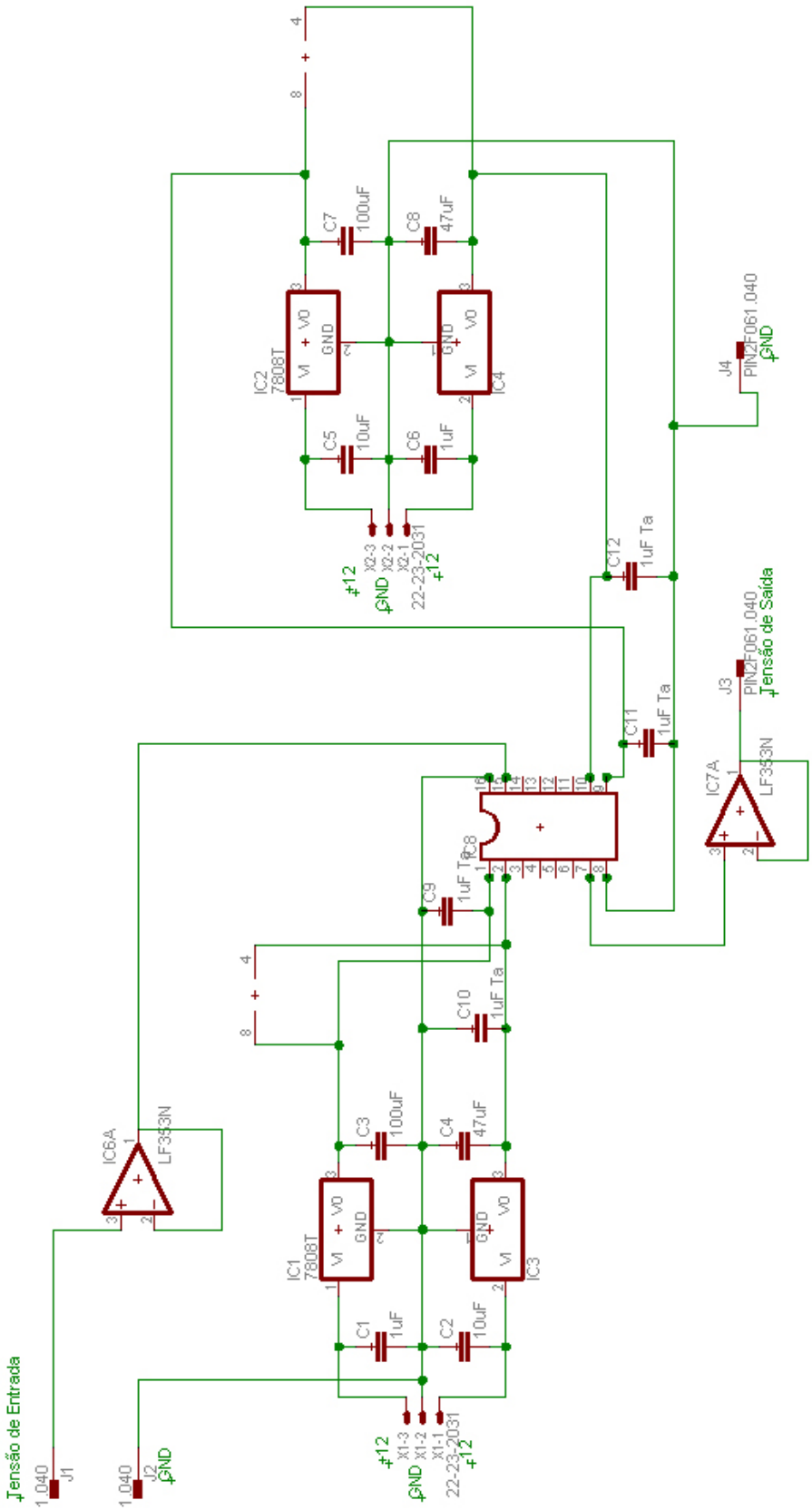

Figura 3.5: Esquemático da placa de condicionamento do sinal de tensão de soldagem

e anulando portanto o isolador.

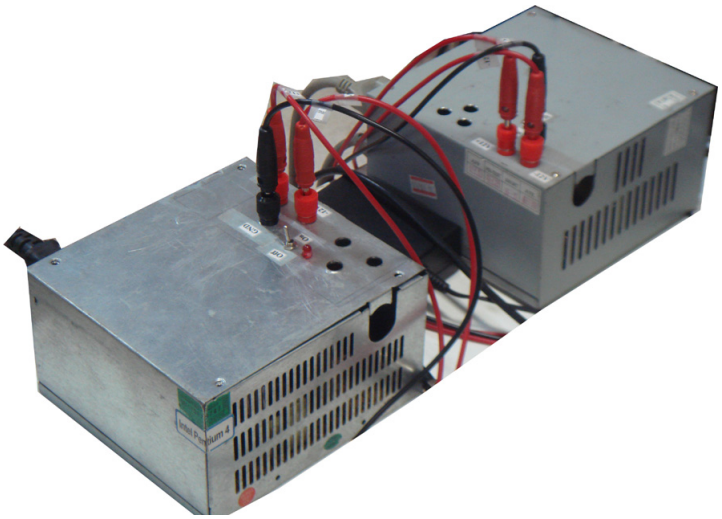

Figura 3.6: Fontes ATX adaptadas para o experimento

O esquemático da placa está na Figura 3.5 em que a tensão de entrada situa-se à esquerda superior e a tensão de saída à direita inferior. Note que existem somente reguladores de tensão modelo LM7808 e o simétrico LM7908 para limitar a tensão de saída do sistema a +-8 V isso significa que utilizando o sistema de redução 10 V para 1 V (presente no sistema de medição original) a tensão máxima de soldagem medida pelo sistema é de +-80 V.

A Figura 3.4 mostra o equipamento montado note as entradas e saídas do sistema e as entradas para fontes de alimentação. Testes realizados na nova placa mostram-se superiores a antecessora com sinal exibido próximo ao original com distorção evidente a partir de frequências superiores a 30 kHz.

3.3.2 Captura de imagens e integração de *softwares* envolvidos

Para iniciar a captura de imagens foi necessário a leitura intensa dos manuais (DALSA-Coreco 2006) do fabricante do *frame grabber* a fim de descobrir algumas características em relação à câmera e como se deve integrá-lo ao software de captura.

No manual de instalação tanto do **frame grabber** quanto do *software* de captura, é imposta a utilização de sistemas *Microsoft Windows*. Para instalar o *frame grabber* tanto o manual quanto o programa de instalação do driver do fabricante exigem a utilização da **API** do fabricante chamada **SAPERA LT 5.2**. Logo que se instala a **API** pode-se instalar o *driver*.

Durante a instalação da **API**, é também instalado um programa chamado *CamExpert*. A partir dele que são testadas e definidas todas as características do processo de captura de imagens como **ROI**, sincronismo, exceto pelo tempo de exposição da câmera que é realizado por um programa *freeware* fornecido pelo fabricante da câmera chamado *PFRemote*.

O programa *CamExpert*, exibido na Figura 3.7, possui em sua base de dados informações intrínsecas sobre câmeras de diversos fabricantes (tipo de câmera, definição). Uma delas, o modelo utilizado nesse trabalho, a câmera **DALSA** modelo *DS-21-001M150* (B) , que se adequadamente instalada (A) na porta *frame grabber* e alimentada, o programa pode iniciar a exibir imagens coletadas pela câmera bastando clicar em **GRAB** (C) e para finalizar a captura clicar em *Freeze* (D). Note em (E) a barra de status indicando a taxa de captura da câmera (20 fps) e a resolução utilizada no momento da captura desse quadro (1024 X 1024).

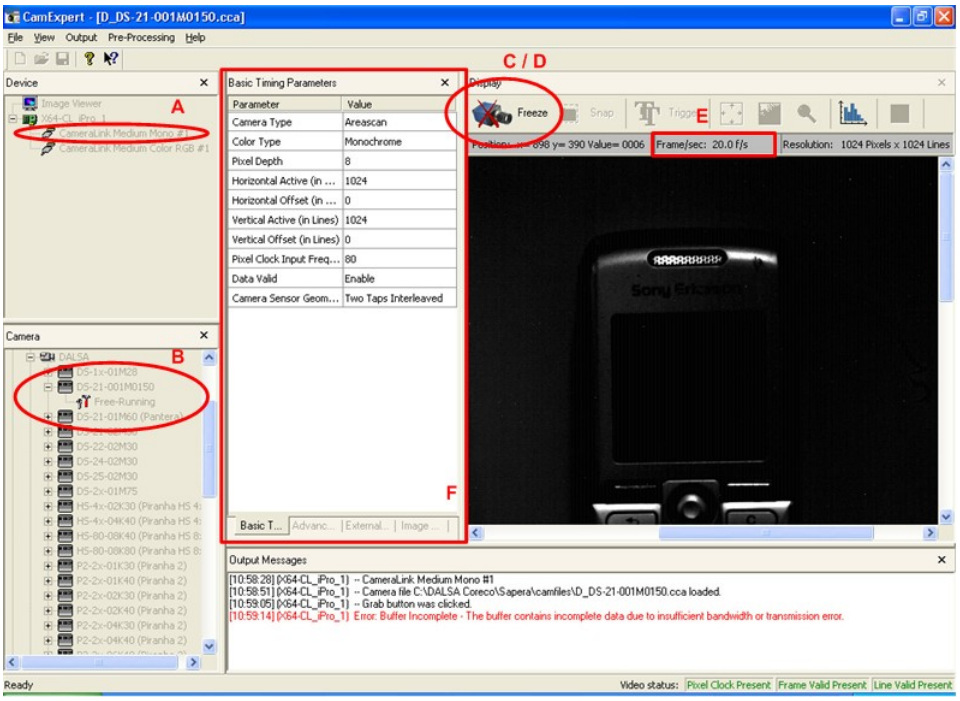

Figura 3.7: Janela do Programa *CamExpert* e processo de captura de imagens

A janela (F) da Figura 3.7 permite ao usuário customizar os parâmetros default da câmera para então adaptá-la a necessidade do experimento. A Figura 3.8 mostra em detalhe a configuração das abas alteradas a fim de serem utilizadas nos experimentos de captura em (A) foi definido uma janela de **ROI** de 128x128 *pixels* deslocada para o centro do sensor de imagem da câmera resultando na imagem (B) para reduzir deformaçãoes radiais e em (C) foi definido o uso do *trigger* associado a uma borda de subida detectada na porta de *trigger* 2 do *frame grabber*.

Figura 3.8: Configuração dos parâmetros como **ROI** e *trigger*

As alterações dos parâmetros da câmera no *CamExpert* podem ser salvas em arquivos de configurção de câmera conhecidos pela extensão do arquivo .ccf, esse pode ser utilizado em conjunto da **API SAPERA LT** para o desenvolvimento de programas propiciando o rápido desenvolvimento do programa, pois evita que o código fonte tenha que ser adaptado a uma série de parâmetros da câmera.

3.3.3 Controle da taxa de captura da câmera - PFRemote

O programa *PFRemote* comunica de forma serial com a câmera utilizando o padrão **CAMERA LINK**. Esse programa permite que sejam configurados diretamente na câmera parâmetros como **ROI** ou múltiplas janelas **ROI**, resolução da captura, correção de iluminação, diferentes formas de sincronismo, calibração, taxa de amostragem e o tempo de exposição da câmera.

Uma vantagem desse programa é a possibilidade de gravar esses detalhes diretamente na memória *flash* da câmera, para que as características definidas se tornem default. Isso pode ser perigoso, pois se realizado de forma inconseqüente, pode-se até impossibilitar a reconexão serial com a câmera. Por isso deve-se ajustar esses parâmetros sem salvá-los na memória *flash* pois toda vez que a câmera for desligada as alterações são perdidas.

A janela inicial do programa *PFRemote* exibe as portas disponíveis e, na placa *frame grabber*, basta escolher onde a câmera foi instalada. A janela da Figura 3.9 mostra onde são configurados o tempo de exposição (A),a taxa de amostragem resultante (B), também é possível alterar o tipo de captura em (C), e o botão para gravar na flash da câmera (D) (um detalhe que ao alterar qualquer parâmetro automaticamente ele é alterado na câmera dispensando o uso de (D)). É possível alterar também o ROI em (E) que ocasiona o aumento da taxa de captura máxima da câmera caso o ROI seja menor que o tamanho máximo da câmera.

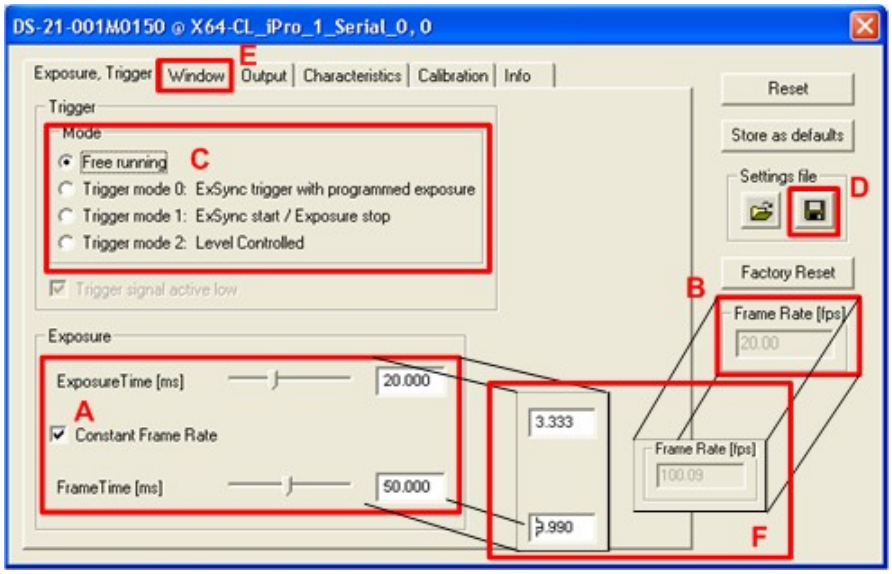

Figura 3.9: Diagrama de configuração do programa *PFRemote*

Ao se ajustarem os parâmetros da câmera como *Exposure Time*, quanto menor o seu valor, maior a taxa de amostragem. Em conseqüência disso mais luz é necessária para se obter uma boa imagem até mesmo para um arco de solda. Porém um alto valor de *Exposure Time* causa um brilho execessivo causando uma imagem saturada (imagem toda branca) podendo até dopar um *pixel* (ficar sem funcionar por um **tempo** imagens com pontos brancos sempre no mesmo lugar).

3.3.4 API SAPERA LT

A instalação da **API SAPERA LT** 5.2 era fornecida pelo CD de instalação do *frame grabber*, para então obter e ajustar a captura de imagens, por meio dos programas *CamExpert* e *PFRemote*. Faltou então o desenvolvimento de um *software* específico para esse trabalho a fim de sincronizar, capturar e analisar a poça de soldagem.

A leitura detalhada dos documentos fornecidos pela **API** o primeiro deles (Dalsa 2006) indicou a existência de três tipos **API** uma voltada totalmente para linguagem *C*, outra voltada para linguagem *C++* com orientação a objetos, e outra voltada linguagens de alto nível utilizando *ActiveX* podendo ser utilizada tanto para *Visual Basic* tanto para *C#* (lê-se c *sharp*).

A princípio, foi proposto desenvolver o software em uma plataforma de um nível mais alto a fim de obter toda a funcionalidade da **API**, por isso foi escolhido o uso do *ActiveX* aplicado a linguagem de programação *C#*. A leitura do manual responsável

(Dalsa 2006) indicou a presença de programas de demonstração instalados junto à **API**, desenvolvidos para *Visual Basic 6.0*, *VB.NET*, *C#* e *Delphi 7*.

Para se desenvolver o programa a plataforma **IDE** (*integrated development environment*) escolhida foi o *Microsoft Visual Studio 2005* por permitir o desenvolvimento de programas em várias linguagens diferentes com *C#*, *Visual Basic*, *C++*, *C* e até *JAVA*.

A partir do programa de demonstração em *C#*, que simplesmente captura e exibe imagens, foi possível inferir o funcionamento básico da **API**. A **API** para funcionar adequadamente ao executar o programa exibe uma janela que pede ao usuário para escolher a porta do *frame grabber* onde foi instalada a câmera e seu respectivo arquivo de configuração gerado pelo programa *CamExpert*.

O manual para desenvolvimento de *ActiveX* (Dalsa 2006) não é muito claro quanto à organização de classes, somente descreve a implementação de partes de código e parâmetros além de evidenciar a preferencia por desenvolvimento em *Visual Basic* misturando várias vezes códigos fontes. Porém o manual (Dalsa 2006) exibe a organização das **API** e das classes reexibido na Figuras 3.10.

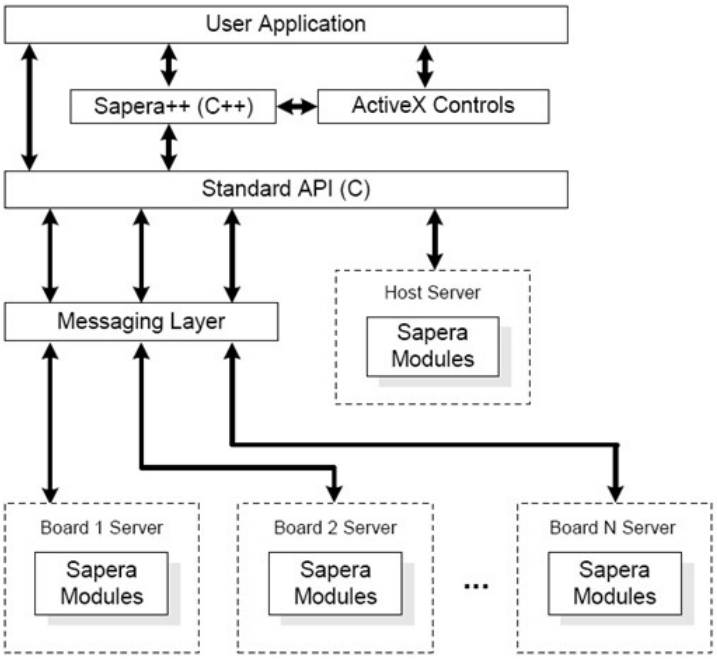

Figura 3.10: Topologia da **API SAPERA LT**

O manual *ActiveX* (Dalsa 2006) especifica bem o uso do programa de demonstração, porém ao adaptar o programa para criar novas classes ou mesmo acessar objetos de-

scritos no manual de demonstração, esses simplesmente não existem ou foram encapsulados de outras formas não descritas no manual. Esses problemas foram apresentados a pessoas mais especializados na linguagem *C#* que sugeriram não utilizar este *ActiveX* até que se torne mais documentado e implementado a essa linguagem.

Ao regredir na topologia da Figura 3.10, optou-se pela **API** destinada a *C++* conhecida como *Sapera++*.

A opção restante, a **API** para linguagem *C* é largamente descrita passo a passo em detalhes no manual (Dalsa 2006) e parâmetros específicos de captura são descritos em (Dalsa 2006), por isso optou-se por utilizá-la. Para usufruir dessa **API** foi utilizada a **IDE** Visual Studio 2005 e para configurá-la e utlizá-la foi criado um documento anexo C que descreve o procedimento instalação e uso.

A grande dificuldade em desenvolver um programa de captura em *C*, é compreender quais são os elementos importantes e como eles se relacionam em (Dalsa 2006) é apresentada a estrutura básica das bibliotecas desenvolvidas em C reexibidas na Figura 3.11.

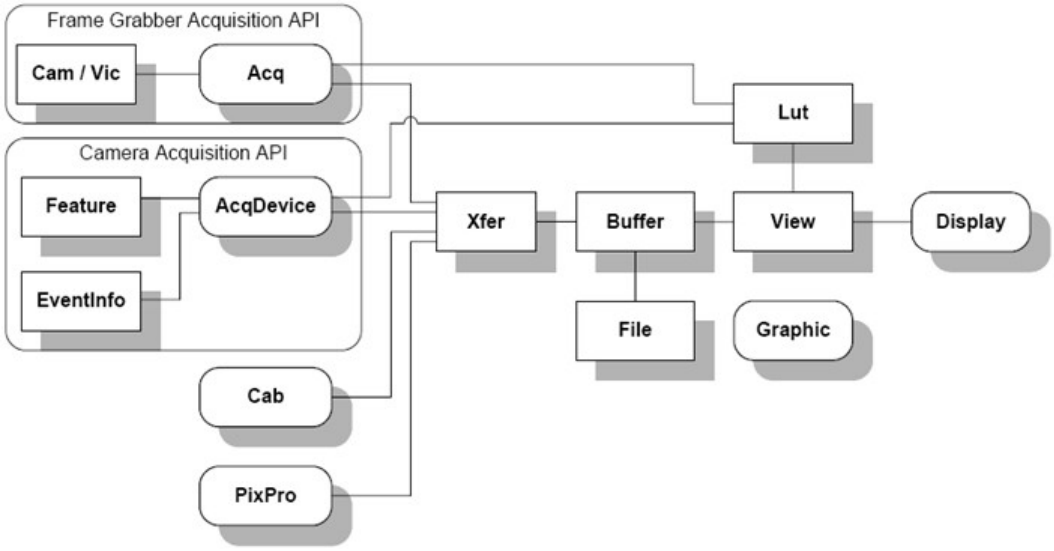

Figura 3.11: Topologia das bibliotecas da **API SAPERA LT** em *C*

A Figura 3.11 possui retângulos bem definidos, os quais são os recursos dinâmicos como dados de imagens e barramentos lógicos. Os retângulos abaulados, são recursos estáticos como a placa de vídeo, o *frame grabber* e a câmera.

No manual (Dalsa 2006) é apresentado um programa de demonstração de captura e exibição básico nas páginas 20-21 (anexo C). Ao comparar a estrutura do programa

com a Figura 3.11 pode-se criar, a partir das variáveis contidas no programa de demonstração, um novo diagrama visto na Figura 3.12 que apresenta o relacionamento das estruturas dinâmicas e estáticas.

Em síntese, o programa de captura e exibição cria 9 variáveis, dentre as quais as 4 mais importantes e independentes são: (a) *hSystem* gerencia variáveis internas do programa *hBuffer*, *hView*, *hDisplay*, *hCam* e *Hvic*; (b) *hAcq* gerencia parâmetros de câmera obtidos em *hCam* e *Hvic*; (c) *hBoard* gerencia os dados resultantes do *frame grabber* e (d) *hXfer* é responsável por interligar e controlar a transferência de dados entre os grupos independentes *hSystem*, *hAcq*, *hBoard*.

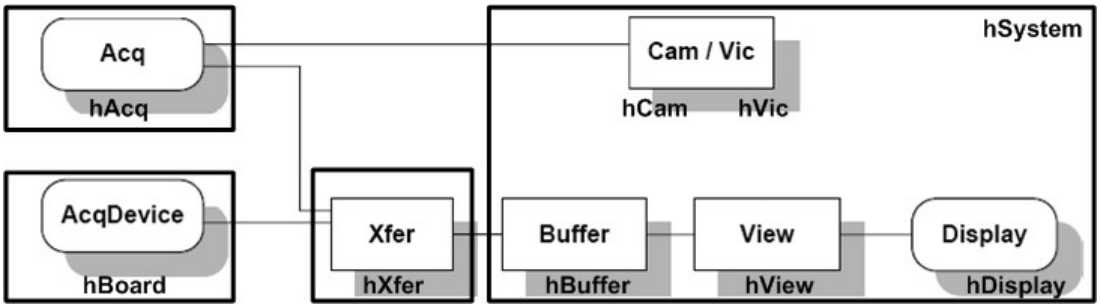

Figura 3.12: Relacionamento, variáveis e estruturas básicas da **API**

Note na Figura 3.12 que a variável *hBuffer* foi acessada somente para exibir imagens, porém, poderia ser utilizada para que alguma função pudesse analisar, processar e salvar em disco. Este ultimo caso é previsto até na Figura 3.11 no relacionamento entre a as variáveis dinâmicas *Buffer* e *File*.

Ao se executar o exemplo proposto do anexo C, novamente ocorreu o mesmo problema das outras **APIs** em linguagens mais evoluídas, alguns comandos previstos na documentação (Dalsa 2006) não existiam como *CorManOpen()* e *CorManClose()*. Porém esse problema dessa vez não afetou o programa.

Todos esses problemas ocorridos nos programas anteriores foram agrupados e, a partir deles, foi criado um chamado ao suporte técnico da produtora da **API** com uma série de sugestões. Após algumas semanas foi liberada uma nova **API**, a **SAPERA LT 6.01** que possuía a resolução de todos os problemas apresentados.

Apesar da melhoria em alguns manuais como do *ActiveX* (Dalsa 2006), mesmo assim manteve-se a linha em desenvolver um programa em linguagem em *C* baseado no programa demonstração que se mostrou eficiente e relativamente simples, com a adição

de um conjunto de adaptações para esse trabalho.

A proposição do *software* básico de captura deveria, a partir de um arquivo de configuração da câmera conhecido, executar uma captura durante um período determinado e resultar na criação de um conjunto de: (a) arquivos contendo em cada um, o quadro capturado pela câmera,(b) um arquivo com um video gerado pelas imagens capturadas,(c) e um arquivo com estatísticas individuais de cada quadro como tempo entre os quadros para detectar a veracidade da taxa de amostragem. Além disso, é proposta uma função de analise digital como intensidade média dos *pixels* sobre o quadro capturado.

A grande vantagem do uso da **API** é a ocultação de variantes de captura que ficam relacionadas no arquivo de configuração de câmera como uso de *trigger* ou alteração na taxa de captura de imagens, isso possibilita que o *software* proposto não seja alterado para cada experimento.

O código fonte do programa proposto é apresentado no anexo D. O algoritmo do programa consiste em alterações importantes no programa de demonstração como a substituição de 1 variável para *buffer* por um vetor com 6000 variáveis *hFilho* armazenados em **RAM** a fim de capturar todos os quadros com eficiência para então serem salvos em disco só ao fim do processo de captura. Além disso, associado a esse vetor de *buffer* foi criado um vetor de tempos em **RAM** para serem também armazenados em disco ao fim do processo. Para aumentar a eficiência foram removidos do programa todos os trechos que envolviam a exibição de quadros capturados.

O principal problema detectado nesse *software* foi o estouro de pilha devido à exigência de grande quantidade de memória **RAM** para os vetores de *buffer* e de tempo, por exemplo 60 segundos a 150 fps exigem 9000, quadros e dependendo da resolução da imagem pode ocorrer com facilidade um estouro de memória mesmo para o computador com 1GB de memória **RAM**. Por isso é imperativo o uso do recurso **ROI** para capturas de longo tempo.

Após a realização de alguns experimentos de captura simples concluem os seguintes resultados: A suposição que a exibição da captura poderia afetar a eficiência do programa é falsa, pois os tempos não foram afetados pela exibição de imagens para capturas em baixa velocidade.

O uso da função de intensidade média foi descartada devido à ineficiência do código (tempos da ordem dos 100 ms). Um detalhe desse programa é a total falta de interação com usuário, pois esse é executado em uma tela preta sem respostas.

Apesar dos problemas apresentados nos parágrafos anteriores, os resultados desse programa se mostraram promissores, pois o mesmo se mostrou eficiente na captura e as estatísticas armazenadas no arquivo de tempo mostraram que os tempos entre os quadros se mantiveram como calculado até para o pior caso, que seriam 150 fps ou seja 6,6 ms.

3.3.5 Melhor forma de capturar a imagem da poça de soldagem

Para facilitar o processamento de imagens pode-se melhorar a imagem capturada por meio de: (a) ajuste do foco de uma objetiva, (b) ajuste do diafragma para maior ou menor entrada de luz, (c) utilização de uma lente de macro a fim de aumentar a imagem (sem deformá-la), (d) uso de filtros ópticos que capturam somente alguns comprimentos de onda (exemplo o infravermelho gerado pela poça de soldagem) e (e) ajuste da posição da câmera a fim de evidenciar algumas nuances da imagem como contornos.

O ajuste dos parâmetros de visualização da poça (escolha de lente, ajuste de foco, abertura do diafragma, tempo de exposição, orientação da câmera) é um problema de otimização multi-variável de difícil solução. Além desses problemas o ambiente onde os equipamentos ópticos estão instalados, no caso da visualização da poça de fusão, é bastante hostil, o que implica na necessidade de se criarem sistemas de proteção do aparato. Para isso, foi proposto a confecção de uma caixa de aço protetora a fim de proteger a câmera contra respingos de soldagem e contra pequenos choques mecânicos. Além disso o suporte foi projetado de modo a permitir o encaixe ajustável da câmera e dos suportes dos filtros, além de uma placa de vidro comum para proteção do conjunto óptico contra respingos.

A Figura 3.13 mostra a caixa protetora da câmera em (A) o rasgo ajustável (oculto) para fixação da câmera, em (B) o rasgo ajustável para o suporte de filtro, em (C) placa protetora de vidro comum, em (D) o encaixe para suporte de tripé.

As objetivas foram encomendadas para encaixarem diretamente na câmera e só existem duas objetivas de grande angular uma de 25 mm e outra de 50 mm, porém existe um pequeno problema quanto ao uso destas o foco descrito na objetiva é o dobro do

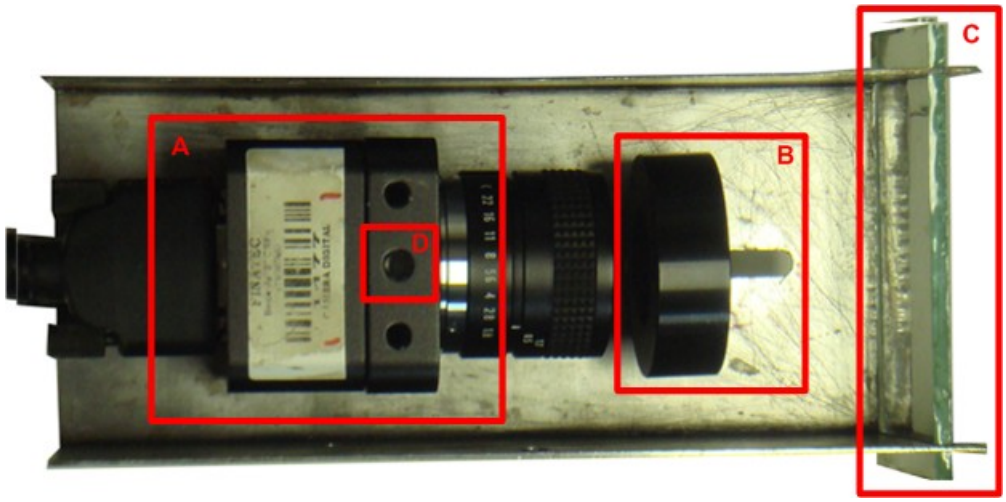

Figura 3.13: Montagem da caixa protetora para câmera

previsto isso significa que o foco mínimo de uma das objetivas é 50 cm e a outra 100 cm dificultando o uso. Por isso, optou-se pela utilização da objetiva de 25 mm ou pelo uso de alguma adaptação por meio de espaçadores, que produzem o efeito de *zoom*, aproximando a imagem e reduzindo a distância da região do foco.

Os filtros ópticos disponíveis na **UnB** foram utilizados em projetos anteriores e já não possuíam nenhuma especificação quantitativa e qualitativa do filtro, devido ao desgaste dos textos nas bordas do filtros. Sabe-se apenas que estes eram de banda estreita na faixa do visível. Outro filtro, emprestado de departamento de física, era um passa alta infravermelho, porém não se sabia a faixa do filtro. A Figura 3.14 mostra os filtros disponíveis.

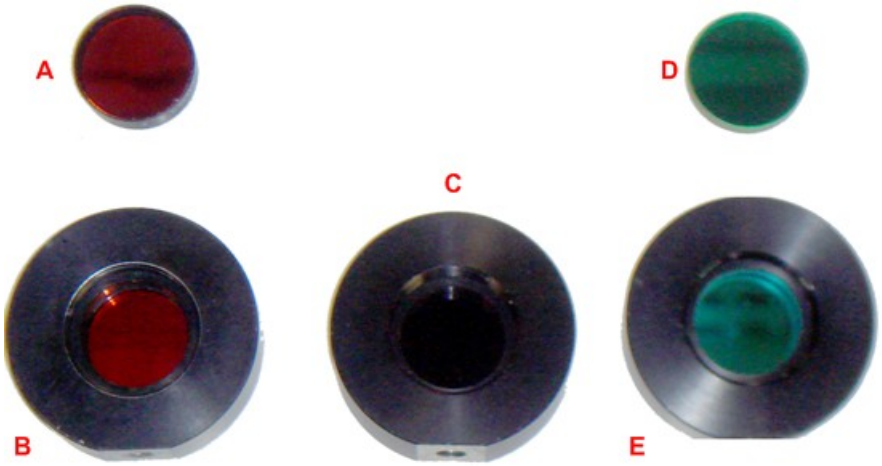

Figura 3.14: Filtros ópticos disponíveis na **UnB**

Por isso foi desenvolvida uma técnica para confirmar o tipo e a faixa do comprimento de onda de forma qualitativa por meio de uso de um espectrômetro também disponível

no **GRACO** procedente de um outro projeto de inspeção de soldagem. Para testar o filtro foi necessário encontrar uma fonte que liberasse energia luminosa em todos os comprimentos de onda e a solução encontrada foi gerar um arco de soldagem por meio de um processo de soldagem do tipo **TIG** (*tungsten inert gas*) restrito a uma base resfriada a água. O diagrama do experimento é mostrado na Figura 3.15.

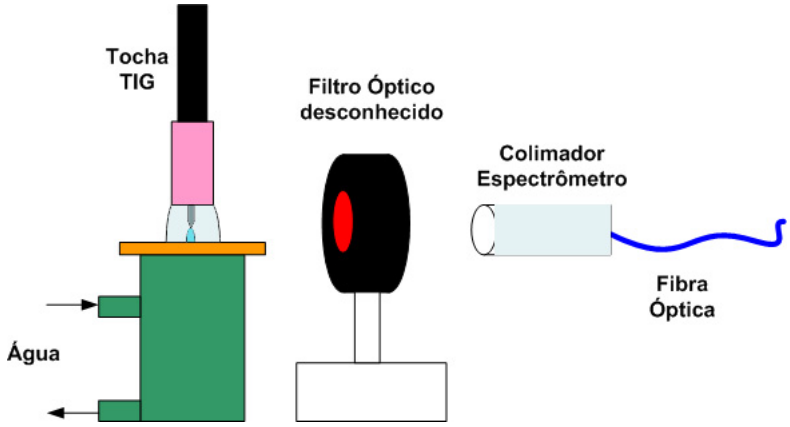

Figura 3.15: Diagrama do experimento para descobrir o tipo e que comprimentos de onda o filtro trabalha

Por meio o programa fornecido pelo fabricante do espectrômetro foi possível inferir o funcionamento dos 5 filtros da Figura 3.14 descritos na tabela 3.1. Os filtros que poderiam ser usados são próximos ao infravermelho baixo por isso os melhores filtros da Figura 3.14 a serem testados nos experimentos são o (A), (B) e (C).

Tabela 3.1: Tabela de tipos de filtros disponíveis

Filtro	Cor visível	Tipo	Comprimento de onda [nm]
A	Vermelho Escuro	Passa Faixa estreito	633 a 636
B	Vermelho Claro	Passa Faixa	624 a 644
C	Infravermelho	Passa alta	acima de 800
D	Verde escuro	Passa Faixa	503 a 520
E	Verde claro	Passa Faixa	380 a 430

Para posicionar a câmera a fim de obter a melhor imagem, deve-se ter em mente o que mais interessa. Dependendo do experimento pode-se optar por um ângulo baixo em relação à peça a ser soldada, o que possibilita a captura de imagens em perfil. Estas podem ser utilizadas para visualização de modos de transferência metálica.

Outra forma é obter um ângulo mais elevado a fim de obter uma imagem da poça como um todo, o que permite a detecção de falhas e de descontinuidades.

3.3.6 Detecção do curto-circuito e sincronização da aquisição de imagens

Para detectar o curto-circuito e sincronizá-lo à captura de imagens, foi necessário desenvolver individualmente os sistemas de medição da tensão do arco de soldagem, descrito em 3.3.1, o sistema de ajuste da captura câmera, descrito em 3.3.2, 3.3.3 e 3.3.5, e o software de captura de imagens, descrito em 3.3.4. Os diversos sistemas integrados a três sistemas de análise da tensão de soldagem diferentes, a fim de que estes sincronizem a captura de um quadro que exiba a poça em estado de curto-circuito.

Os três sistemas de análise de tensão foram baseados em equipamentos diferentes. O objetivo é definir maneiras eficientes de analise das formas de onda produzidas pelo sistema de medição, para que seja possível detectar e sincronizar um software de captura de imagens.

O sistema de medição da tensão do arco de soldagem utilizado nesse trabalho produz sinais analógicos. Logo para que um computador possa obter dados digitais, esse deve possuir algum equipamento que capture a tensão fornecida pelo medidor e a converta em um sinal digital e apresente esse dado a algum software que possa usar esses dados.

Para isso existem no laboratório do **GRACO** 3 equipamentos de captura: um do fabricante **EAGLE DAQ** da *National Instruments* e uma placa microcontrolada produzida no **GRACO**.

O primeiro é uma placa **PCI** modelo **PCI-703S-16A** que possui a capacidade de capturar 16 canais analógicos simultâneamente a uma taxa de 400 kHz com duas saídas **D/A** (digital /analógica), todas com 14 bits de resolução, mais 8 canais digitais TTL. O fabricante disponibiliza com a placa kits **SDK** (*software development kit*) para *Borland C Builder 5* e para *LabView*, essa placa também é utilizada no controle da máquina de soldagem.

O segundo equipamento de aquisição não é uma placa **PCI** mas uma pequena caixa que pode ser conectada a um computador via porta **USB** modelo **NI USB-6009**. Este é um sistema de aquisição de baixo custo que possui 8 canais analógicos multiplexados a uma taxa máxima de 48 kHz de 14 bits além de 2 canais **D/A** e 12 canais digitais. Esse equipamento possui um problema: por ser multiplexado, a taxa máxima de captura é dividida entre o número de canais de capturados. Logo capturam-se 3 canais para obter taxas máximas de 16KHz por canal. O fabricante fornece **SDK** para *LabView* e *Visual Studio.NET*.

O terceiro equipamento na verdade não é um equipamento dedicado à aquisição de dados, mas um microcontrolador do fabricante **MICROCHIP** família PIC18F452 onde se utilizaram as portas **AD** a fim de colocar todo o software de análise da tensão em um só equipamento.

A idéia de utilizar três equipamentos diferentes é a de definir a forma mais eficiente, com os equipamentos disponíveis , de detectar e transmitir o sinal de sincronismo para o *frame grabber* a fim de reduzir o atraso entre a detecção e a captura do quadro.

Para implementar os testes, foram desenvolvidos códigos fontes em *LabView* e em *C*, que possuíam em sua estrutura a mesma rotina diferenciada somente pela forma como o sinal de sincronização é transmitido ao software de captura. Os *softwares* em *LabView* transmitem o sinal de sincronização via rede *ethernet* por meio do uso de *sockets* e o *software* desenvolvido em *C* para microcontroladores transmite diretamente o sinal em um canal de sincronização dedicado do *frame grabber*

O algoritmo para detectar curto-circuito é inspirado na técnica de janelamento descrita em 2.3. O software deve possuir dois processos em paralelo: (a) um para obter uma janela suficientemente grande e calcular e atualizar os limiares máximo e mínimos de tensão do arco; (b) analisar o sinal de tensão para detectar transições de sinal entre limiar de máximo e um limiar mínimo para gerar um sinal de pulso de sincronização (veja a Figura 3.16).

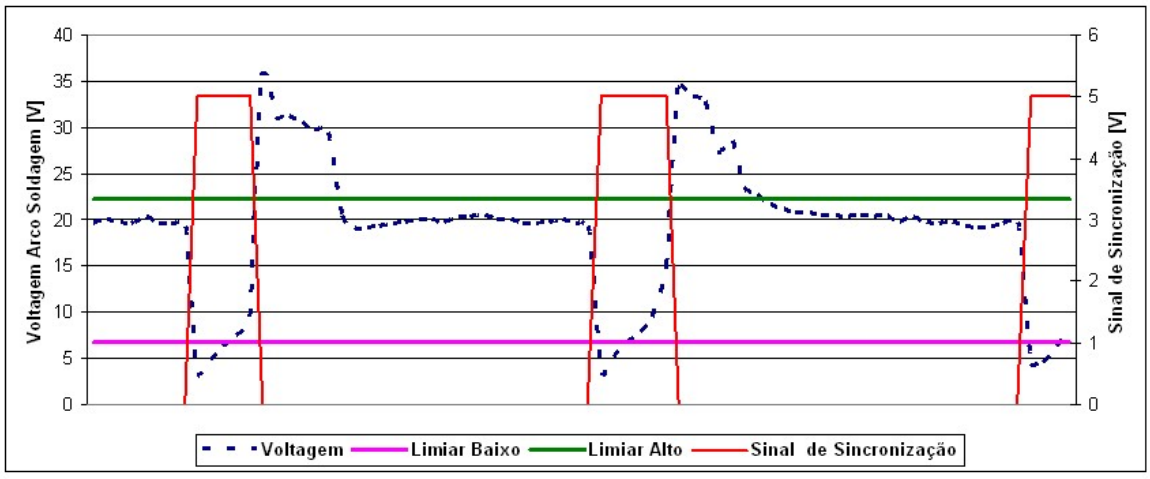

Figura 3.16: Gráfico de detecção e sincronização entre o curto-circuito e a captura de imagens

Para definir os valores limiares default foram realizados experimentos de soldagem **GMAW** por curto-circuito padrão a fim de capturar a forma de onda do arco de soldagem para posterior processamento. A Figura 2.5 o mostra resultado da tensão do

arco de soldagem padrão fornecido pela máquina de soldagem, capturada pela placa de aquisição **PCI-703S-16A** e filtrada via software. Note que os valores exibidos na Figura 2.5 exibem a tensão do experimento que foi calibrada experimentalmente para placa **PCI-703S-16A**. Essa tensão foi novamente revertida para a tensão real da máquina obtendo os limiares máximo e mínimos.

Para implementar os testes no *software LabView* nas placas **NI USB-6009** e **PCI-703S-16A** foi definido um *software* mudando apenas parâmetros para o funcionamento de cada **VI** (*vitrual instrument*) fornecido pelo fabricante do equipamento.

O algoritmo consiste na abertura de um canal de comunicação entre o *software* de captura de imagens e o de análise de tensão. Quando o *software* de captura de imagens inicia a captura, o programa de detecção de curtos é iniciado. Toda vez que o programa detecta um curto-circuito, ele envia um caracter para que o software de captura marque o quadro capturado para ser processado. Isso ocorre enquanto não houver uma interrupção fornecida pelo software de captura de imagens.

Para implementar os testes com a placa microcontrolada, foi utilizada uma placa do um projeto de graduação (Franco 2005) que possui dois conectores do tipo RJ45 (conector de rede *ethernet*) que foram utilizados para ligar ao sistema de medição da máquina de soldagem e outro para ligar o canal de sincronização diretamente ao microcontrolador.

O algoritmo é mais simples. Uma aplicação analisa variações na onda e gera um pulso diretamente na porta de sincronização. A vantagem é que não houve mudança alguma no software de captura de imagens somente a alteração do arquivo de configuração de câmera exibido na Figura 3.8 (C).

Para testar as três formas de sincronização, primeiramente foram simulados sinais de soldagem senoidais ligados a cada sistema de detecção de soldagem.

Para testar a placa **PCI-703S-16A** foi conectada a porta 2 e a placa foi configurada para capturar a uma taxa de 400 kHz. Os resultados desse experimento não foram satisfatórios, pois o sistema análise de tensão apesar de estar configurado para trabalhar a 400 kHz não trabalhou muito bem.

O porquê do software de captura não operar adequadamente está relacionado com a requisição de uma única amostra instantânea para detecção de variações de onda, isso

causou ineficiência no código. Os tempos registrados no arquivo de tempo do software de captura de imagens que deveriam ser em torno de 10 ms (a onda senoidal era de 100Hz) eram inconstantes e incoerentes com tempos entre *frames* de 50 ms e 30 ms.

O mesmo problema ocorreu com o uso do outro equipamento de captura, **NI USB-6009**, em que os tempos entre os quadros foram ainda piores em torno de 200 a 300 ms.

O problema desses equipamentos é que eles são desenvolvidos para serem *data loggers*, ou seja, equipamentos de monitoração de eventos baseados em janelas. Deve-se preencher uma janela com dados de captura para então repassá-los ao software de análise de tensão. Os equipamentos funcionam bem para captura de dados mas não são desenvolvidos para capturas pontuais em tempo real.

Por exemplo, a janela de dados da placa **PCI-703S-16A** é 4096 amostras de 14 bits. Para completar essa janela é necessário, a uma taxa 400 kHz, um tempo mínimo teórico de 10,24 ms. Outro problema é a possibilidade de que a implementação de *sockets* em ambos os softwares talvez não seja eficiente.

Já para o teste utilizando o microcontrolador que realizava captura e analise de uma amostra a cada 50 us (20kHz), foram obtidos bons resultados para freqüências senoidais de até 1KHz vistas em um osciloscópio. A partir dessa frequência, existe um atraso de fase entre o sinal obtido e o pulso de curto-circuito.

Devido à impossibilidade de se utilizarem os sistemas de captura de dados tanto da fabricante EAGLE, quanto o fabricante **NI**, optou-se pelo uso do sistema microcontrolado. A desvantagem desse sistema é que, da forma como o a placa de circuito está montada, não é possível colocar uma tensão de referência negativa, ou seja, o sistema só pode ser conectado em ambientes em que as tensões possam variar de 0 a 5V

3.4 EXPERIMENTO BÁSICO

A partir de testes individuais, foi proposta a montagem de todos os sistemas: de medição do arco, de determinação e sincronização de captura, de captura de imagens, de suporte ao equipamento óptico, câmera, objetiva e filtros, para que, em conjunto, permitam a obtenção da melhor imagem possível de uma poça de soldagem fundida.

Para ajustar e familiarizar com o conjunto de sistemas montados foi proposto primeira-

mente capturar imagens do processo de soldagem **GMAW** por curto-circuito a fim de ajustar parâmetros como: (a) a distância mínima entre o suporte de câmera e o experimento de soldagem; (b) o foco da objetiva e abertura do diafragma; (c) determinar se realmente o uso de filtros é eficaz; (d) determinar melhor taxa de captura; (e) determinar o tamanho de uma janela ROI e (f) verificar como o software de captura integrado ao sistema de determinação do curto-circuito trabalha com sinais reais de soldagem.

No período de realização dos testes, alguns elementos da estrutura ainda não estavam finalizados ou mesmo confiáveis como: a mesa linear e as melhorias nos sistemas de medição, por isso em substituição a esses equipamentos foi proposto o uso do braço robótico para simular a mesa linear e o sistema de medição do arco de soldagem utilizado foi o fornecido pela própria máquina de soldagem.

Para controlar a máquina de soldagem foi utilizado o programa de controle em *LabView* desenvolvido especificamente para a máquina **IMC-450** que só trabalha com uma placa de controle e aquisição desenvolvida no **GRACO** e integrada ao uso com a placa de aquisição de dados **PCI-703S-16A**.

Para realizar os experimentos de captura de imagens foram definidos parâmetros de soldagem padrão **GMAW - MAG** aplicado a todos os experimentos exceto pelo substrato de soldagem. O gás de proteção é uma mistura de argônio com 20% de CO_2 viberado a uma vazão de 12 l/minuto. Para se obter um modo de transferência metálica por curto-ciruito foi ajustada uma tensão de 20V para um arame cobreado aço-carbono de 1 mm de diâmetro, mantendo a tocha de soldagem com um *stand off* de 15 mm de altura. O braço robótico se moveu em linha reta a uma velolcidade de 4 mm/s.

Durante a realização de ajustes, os primeiros experimentos realizados não utilizaram sistemas de sincronização captura, somente a livre captura com taxa de amostragem limitada ao tipo de experimento. A Figura 3.17 mostra o diagrama completo de todo o experimento.

O primeiro experimento proposto consiste na captura de imagens utilizando o programa *CamExpert*, a fim de ajustar o foco e alterar a taxa de amostragem sem inciar o processo de soldagem. Percebeu-se que para ajustar o foco foi necessário colocar uma folha branca impressa com textos, a fim de obter uma região de foco próxima à tocha de soldagem. Percebeu-se que ao aumentar a taxa de amostragem mais escuro fica o objeto em foco pois menos luz é captada pelo sensor de imagem. Por isso optou-se por

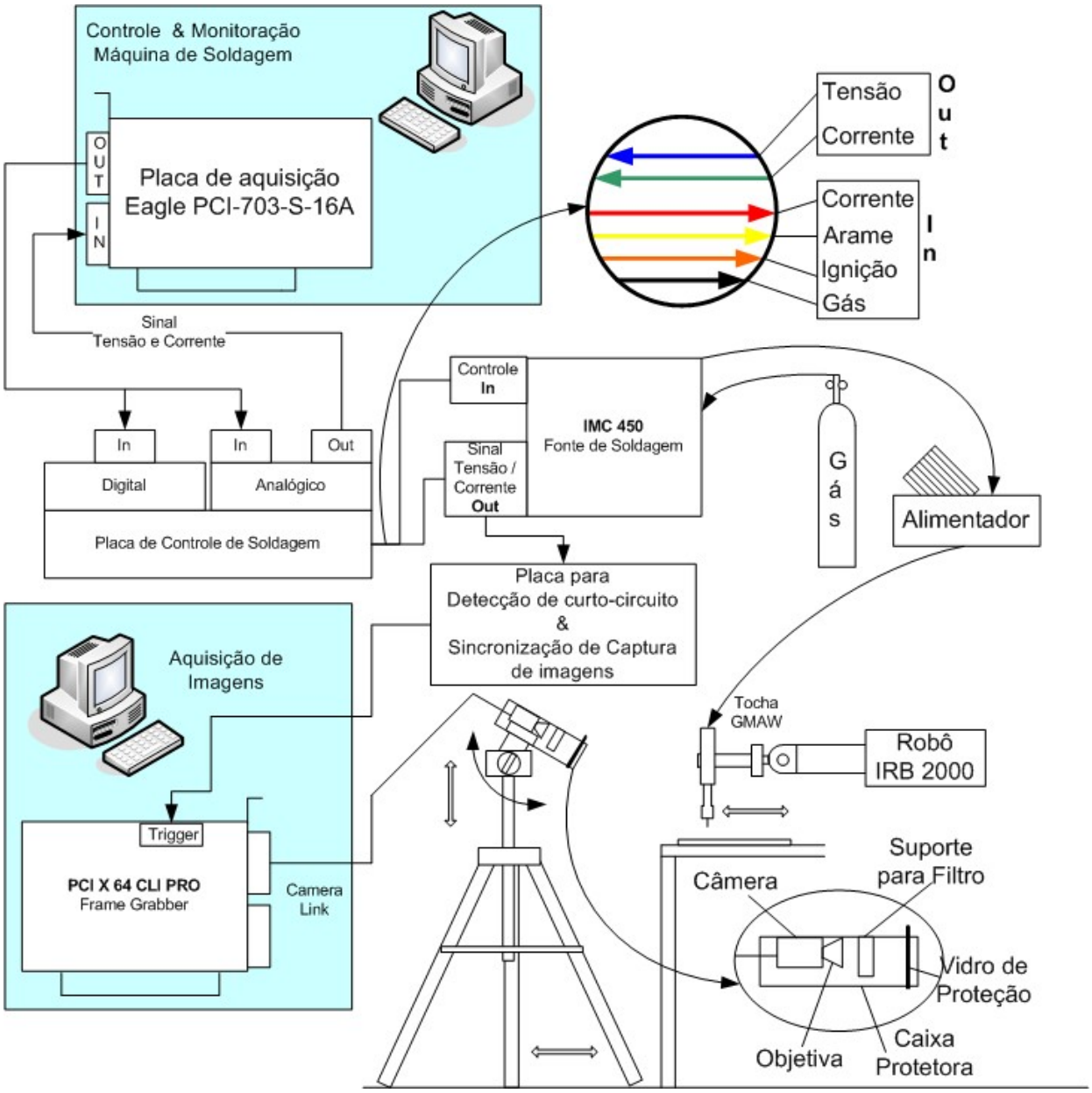

Figura 3.17: Esquemático completo do experimento básico

deixar o diafragma da objetiva totalmente aberto para captura de imagens.

O segundo experimento foi capturar durante o processo de soldagem quadros utilizando ainda o programa *CamExpert* e sem o uso de filtros ópticos, a fim de ajustar a taxa de amostragem. Percebeu-se que para taxas de aquisição muito baixas, menores que 100 fps, o quadro se torna todo branco, e para taxas mais altas a imagem obtida é mais escura.

O terceiro experimento deu-se com a utilização de filtros ópticos disponíveis, inclusive os não indicados em 3.3.5, utilizando taxas de amostragem superiores a 100 fps. Como a maioria dos filtros são passa banda, exceto pelo filtro infravermelho, percebeu-se que para filtros tendendo para ultravioleta (figura 3.14 (D) e (E)) evidenciam os fumos produzidos pelo processo de soldagem, enquanto filtros tendendo para o vermelho (Figura 3.14 (A), (B) e (C)) evidenciam a poça de soldagem e os respingos. Os melhores resultados foram obtidos utilizando os filtros vermelhos.

Uma observação quanto ao terceiro experimento é que o filtro passa alta infravermelho (Figura 3.14 (C)) para tempos de exposição muito baixos, formam imagens sem definição (desfocadas), mesmo que a objetiva estivesse em foco, suspeita-se que o problema seja causado pelo reduzido tempo de amostragem associado ao comprimento de onda restringido pelo o filtro. A restrição limita a faixa de comprimentos em que do sensor de imagens captura, por isso as imagens são obtidas com uma mínima quantidade luz.

A partir das figuras capturadas no terceiro experimento foi possível determinar o tamanho do quadro **ROI** a ser utilizado nos experimentos que está na ordem de 128 X 128 *pixels*. Essa característica foi salva em um arquivo de configuração para o quarto experimento utilizando o *software* de captura de imagens desenvolvido em 3.3.4.

O quarto experimento utiliza o *software* de captura de imagens com o aquivo de configuração produzido no experimento anterior. O *software* foi configurado para capturar 10 segundos de soldagem e salvar todos os quadros capturados e o arquivo de tempos, sem o uso de sincronização.

Os resultados desse experimento mostram-se bons, pois foi possível obter em detalhes cada quadro obtido. Isso não é possível de ser realizado com programa *CamExpert*. Além disso foi possível visualizar imagens de períodos em que realmente era possível

obter somente a poça de soldagem.

Garantido o funcionamento do programa de captura de imagens foram repetidos todos os experimentos com os filtros ópticos a fim de se obterem melhores imagens.

O quinto experimento foi sincronizar a captura de imagens à presença de curto-circuito utilizando a placa desenvolvida em 3.3.6. Essa a placa foi ligada diretamente ao sinal da tensão do arco fornecido pela máquina de soldagem. Um possível problema em se ligar diretamente esse equipamento à fonte de soldagem é possibilidade de que as tensões produzidas sejam maiores que as suportadas pelo microcontrolador, ou seja, pode-se queimar o equipamento. A fonte utilizada produzia sinal de tensão em circuito aberto da ordem do valor ajustado na máquina, que resultava em um valor de aproximando de 2 V após passar pelo divisor de tensão. A falta de um sistema de proteção limita o uso do equipamento a processos de soldagem com tensão positiva de, no máximo, 50 V.

Para realizar esse experimento foi necessário alterar o arquivo de configuração da câmera para capturar imagens via pulso de sincronização na porta de sincronização 2, além disso, ainda é necessário configurar a maior taxa de captura utilizando o programa *PFRemote*. Pela experiência dos experimentos anteriores as melhores imagens foram obtidas entre 120 e 140 fps fora desse intervalo as imagens eram muito claras ou muito escuras respectivamente.

Sabe-se que a câmera utilizada nesse trabalho pode capturar no melhor caso 150 fps em sua configuração original ou seja pode capturar uma imagem a cada 6,6 ms e no experimento em questão foi ajustada uma taxa de de 130 fps ou seja 7,7 ms entre quadros.

Os resultados não foram satisfatórios, pois foram obtidos vários quadros de imagens com brilho excessivo, além disso os tempos entre quadros , em grande parte, ficaram muito próximos à taxa de captura máxima do experimento.

O resultados apresentados foram influenciados, principalmente, pelo tempo de exposição muito longo do sensor de imagens associado a períodos entre curtos muito curtos, pois segundo (Norrish 1992) o período entre os curto-circuitos é baseado em uma distribuição gaussiana com média em 7 ms e desvio chegando a 1 ms.

Conclui-se que existe um período muito longo entre quadros em que é possível a ocorrência de novos curtos entre quadros, isso influencia na captura de imagens em curto, com pouco brilho ou com muito brilho. O problema pode ser resolvido por meio da alteração das configurações da câmera para obter uma maior taxa de amostragem.

A estrutura montada nesses experimentos apresentou um problema na obtenção de imagens relacionado ao suporte da câmera, pois este estava fixo em relação ao movimento do braço robótico e devido a isso existia somente um pequena região durante o experimento que era possível obter uma imagem em foco, esse problema foi resolvido com o uso da mesa linear desenvolvida em 3.2 apresentando as mesmas conclusões descritas anteriormente.

Os resultados e experiência adquiridos nos experimentos anteriores indicaram a necessidade de melhorias em quase todas as partes do sistema de sincronização, captura e análise da poça de soldagem **GMAW** fundida, para isso, foram finalizadas parte da infra-estrutura inacabada, proposição de mudança na montagem dos experimentos, e melhoria no software de controle do processo de soldagem, sincronização e captura de imagens.

A nova montagem proposta foi tentar ao máximo, fixar a câmera em posições de referência, pois ela é a parte mais leve e delicada de todos os componentes. Além disso a mesa linear e tocha de soldagem possuem um considerável peso o que provém uma certa inércia ao conjunto. A fim de fixar a câmera uma pequena adaptação na caixa protetora da câmera foi realizada para que a mesma fosse adaptada diretamente a um braço robotizado ou a um suporte, para então, permitir a desejada repetibilidade.

A tocha de soldagem foi fixada em um suporte já utilizado em projetos anteriores. Esse permite o ajuste de razoável precisão na altura e do ângulo de soldagem. Note que o conjunto tocha-câmera deve ser fixado e referenciado em relação a posição da mesa linear.

Os *softwares* utilizados nesses primeiros experimentos obtiveram resultados insatisfatórios, pois não foram implementados da melhor forma, isso afetou o software de detecção de curto-circuito que não utilizou ao máximo a técnica de janelamento e não foi possível implementar uma função eficiente de análise de imagens. Por isso foi revisado o software de detecção de curto-circuito e foram estudadas **API** de análise eficiente de imagens.

3.4.1 API OpenCV

Um dos problema apresentados pelo sistema de captura e análise de imagens foi a impossibilidade de implementar um software de analise das imagens causado, principalmente, pela forma como foi acessada a variável que contém a imagem capturada.

A **API SAPERA LT**, quando aloca na memória uma imagem, ela na verdade não aloca uma matriz que se assemelhe a uma imagem bidimensional, mas um único e longo vetor contendo uma imagem. As primeiras tentativas em utilizar esse vetor da forma tradicional foram infrutíferas, principalmente, pela falta de experiência profunda em ponteiros e a falta de exemplos na documentação sobre o assunto.

Foi pesquisado uma forma de processar imagens de forma eficiente e foi encontrado uma **API** *OpenCV*, esta foi desenvolvida pela *INTEL* e gratuitamente distribuída pela *internet* (`http://sourceforge.net/projects/opencvlibrary`). Seu intuito é propiciar uma ferramenta de processamento de imagens que contenha a maioria das funções conhecidas na área de visão computacional (500 funções) , além disso, é eficiente, por isso, voltada para ambientes de programação em *C/C++* para aplicações em tempo real (Pisarevsky 2007).

A grande vantagem é a possibilidade de dispensar o uso de ferramentas de alto custo como o *MATLAB* ou o *Halcon*. Outra vantagem é a distribuição gratuita, além disso existe uma comunidade para criar exemplos, reportar problemas e a própria instalação da **API** possui série de programas de demostração.

Primeiramente foram estudados exemplos de uso da **API** e de como ela funcionava. Para instalar a **API** existem passos descritos até em português em
((Ribeiro, A. e Figueiredo 2005)). Para estudar seu funcionamento foi analisado um programa que aplique um filtro do tipo *Canny Edge* para detecção de bordas disponível na instalação
(`OpenCV\samples\c\edge.c`).

A análise do programa conclui que realmente se trabalha de forma eficiente com ponteiros, por meio do uso de *struct* de ponteiro (forma aninhada de armazenar vários tipos de dados em uma só variável). Outro detalhe importante no exemplo é que **API** *OpenCV* trabalha com vetores assim como a **API SAPERA** o que facilita a integração das **API**´s em um só programa.

Por meio da leitura de um manual introdutório em (Agam 2006) sobre **API** *OpenCV* foram obtidos as informações mais importantes como a estrutura da variável de armazenamento de imagem nesta **API** e como acessar de forma eficiente um ponto de um quadro de imagem.

A estrutura da variável de imagem é descrita por várias sub variáveis que permitem armazenar os dados da imagem, e as características da imagem como altura, largura, número de canais de cores, espaço ocupado, tamanho da linha e outras. As principais sub variáveis são descritas na tabela 3.2 :

Tabela 3.2: Tabela da variável de imagem *IplImage*

IplImage		
	int nChannels	Número de canais de cores (se em cores **RGB** = 3 se cinza = 1)
	int depth	Nível de tons de um pixel (8 Bits)
	int width	Largura (*pixels*)
	int height	Altura (*pixels*)
	char imageData*	Dados da imagem em um vetor único
	int origin	Ponto inicial (1,1) incia no canto superior esquerdo ou o contrario
	int widthStep	Tamanho da linha da imagem
	int imageSize	Tamanho da imagem em bytes para alocação dinâmica

Segundo (Agam 2006), para acessar de forma eficiente um ponto (i, j) de uma imagem armazenada em uma variável do tipo *IplImage*, deve-se utilizar a sub variável *widthStep* a fim de posicionar o ponteiro da imagem associando i, j e *widthStep* para então acessar o dado de imagem armazenado no vetor *imageData*, veja o exemplo onde a variável do tipo *IplImage* é chamada de *img*:

```
((uchar *)(img->imageData + i*img->widthStep))[j]=254;
```

Para integrar os dados de imagem da **API SAPERA** a **API** *OpenCV* basta copiar o conteúdo do vetor da variável onde está armazenada a imagem na **API SAPERA** para a variável de imagem da **API** *OpenCV* na sub variável *imageData*.

Para que isso ocorra corretamente, deve-se primeiramente criar uma variável da **API** *OpenCV* vazia com as mesmas características de imagem definidas no arquivo de configuração de câmera. Um detalhe importante é que para as duas variáveis trocarem dados de forma coerente elas devem possuir a mesma altura, comprimento, profundidade de *pixel* e número de canais de cores.

A integração das **API SAPERA** a **API** *OpenCV* abre uma infinidade de possibilidades para que, por meio da **API** *OpenCV*, sejam desenvolvidos funções eficientes para o processamento de imagens e, por meio da **API SAPERA**, obtenha uma captura de imagens eficiente.

3.4.2 Proposição de um programa de medição da poça de soldagem

Para produzir e testar um conjunto de funções que permitam o processamento de imagens foi proposto o desenvolvimento de um *software*, que a partir de imagens obtidas em experimentos anteriores, seja capaz de detectar um quadro que contem um curto-circuito com pouco ou nenhum brilho para então medir a poça de soldagem em dois itens: a largura, e a posição do arame da poça de soldagem para então gravar esses dados em arquivo. Esse *software* será chamado de analise de imagens *offline*

O objetivo desse software é utilizar somente a **API** *OpenCV* para definir, estudar e desenvolver um código que possa detectar: (a) uma imagem com pouco brilho; (b) aplicar filtros para retirada de dados desnecessários; (c) calcular a largura da poça e posição do arame de soldagem por meio do uso de primeira e segunda derivadas proposta em 2.4.5.

Antes de implementar o *software* diretamente foi proposto validar parte o algoritmo a ser implementado primeiramente no *MATLAB* utilizando, somente, imagens pré-selecionadas a fim de validar o algoritmo de filtro e medição da poça de soldagem (o código fonte está no anexo F).

Os resultados relacionado ao uso de filtros mostram que é necessário o uso de um tipo de filtro digital do tipo *threshold*, pois existe um ruído de fundo também previsto e evidenciado na Figura 2.12, além disso, existe a necessidade de retirar um vulto que pode aparecer causado principalmente por exposição excessiva do sensor de imagem um exemplo é Figura 3.19 (B) e (C).

O sensor de imagens ao capturar valores baixos próximos a zero (cor mais escura) evidencia o ruido de fundo inerente ao processo de captura de imagens. Para reduzir ou pelo menos retirar o ruído do fundo de uma imagem de uma poça de soldagem deve-se levar em conta o elevado contraste ente o fundo e a poça de soldagem. De forma empírica, ao se analisar esse contraste em diversas imagens, nota-se que os pontos do fundo possuem uma intensidade luminosa variando de 0 a 20 e os pontos dos valores da poça variam entre 100 e 255.

Por isso para retirar o ruído de fundo basta definir um limiar para intensidade luminosa do *pixel* no valor de 20 para transformar o fundo da imagem no valor zero. A Figura 3.18 mostra um exemplo em (A) a poça em formato original e em (B) a poça filtrada, note em (B) que até os respingos de soldagem foram mantidos.

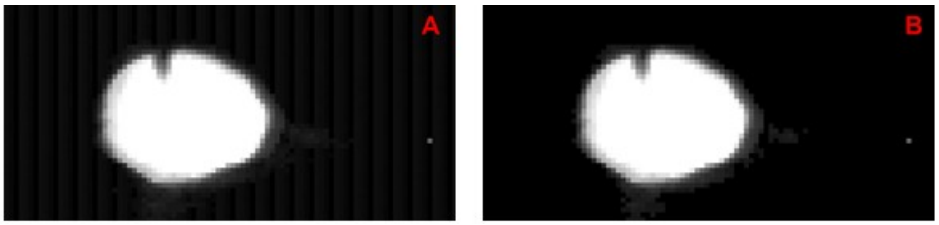

Figura 3.18: Aplicação do filtro *threshold* a fim de retirar o ruído de fundo

Para retirar os vultos de uma imagem, já sem o ruído de fundo, foi proposto a reaplicação de um filtro de *threshold* porém com um limiar calculado por meio do brilho médio, onde somente os pontos de intensidade do pixel maiores que zero são computados na média. A Figura 3.19 contém a análise de várias poças com diferentes intensidades de brilho e o resultado dos filtros de ruido de fundo e vulto.

Note que não foi possível, em todos os casos, obter o melhor resultado filtro de vulto, pois esse filtro é eficiente até certo ponto de intensidade de brilho, por isso foi proposto anteriormente a utilização de uma função que detectasse imagens de curto-circuito e com pouco ou nenhum brilho do arco somente.

Outra forma de se identificar a melhoria da qualidade de imagem é observar os gráficos relacionados à soma das colunas da imagem comparando a Figura 2.12 com o resultado do filtro de ruído de fundo aplicado em outra poça de soldagem na Figura 3.20.

Para medir parâmetros da poça de soldagem como largura da poça e posição do arame com base no artigo (Koike 1999) e aproveitando do alto contraste entre o fundo e a poça fundida, foi implementado um sistema de aferição que utiliza primeira e segunda derivadas associadas a algoritmos de localização de valores máximos e mínimos locais para determinar o início e o fim da borda da poça em relação ao fundo e além disso, determinar a posição do arame. Essas variáveis possibilitam a determinação da largura da poça e da centralidade do arame em relação a poça.

A Figura 3.21 mostra dois gráficos sobrepostos: o somatório das colunas da imagem da poça (em azul) e sua primeira derivada (em verde), note que as posições das bordas (linhas em vermelho) podem ser determinadas por meio dos pontos de inflexão do

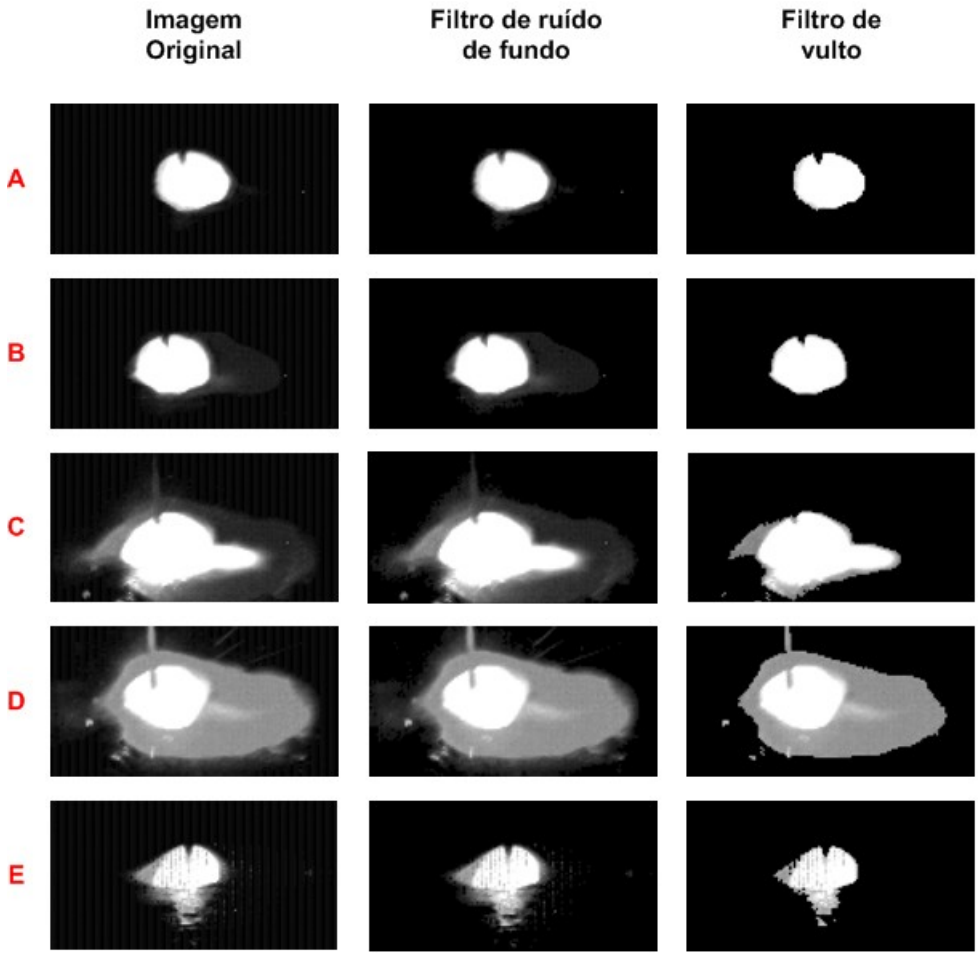

Figura 3.19: Funcionamento dos filtros de ruído de fundo e vulto em diferentes imagens

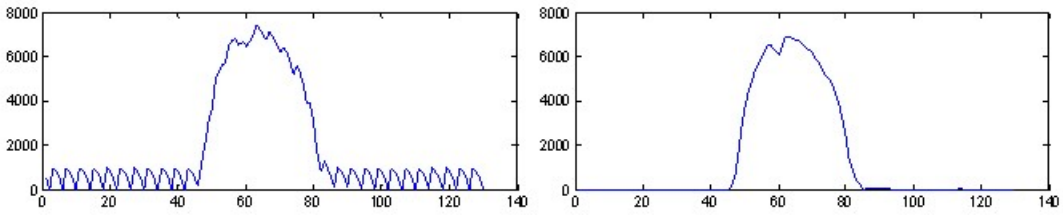

Figura 3.20: Resultado grafico da retirada do ruído de fundo

gráfico que são máximos e mínimos locais. Note que não existe ponto maior ou menor que os delimitados pelas linhas azuis horizontais (os máximos e mínimos locais).

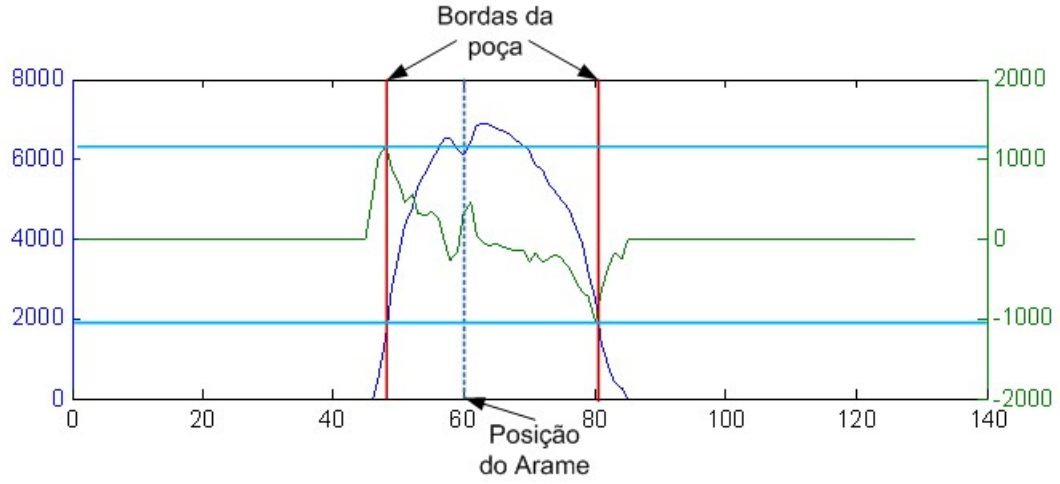

Figura 3.21: Identificação da posição do inicio e fim da poça e da posição do arame de soldagem

Pode-se notar também que a posição do centro do arame pode ser relacionada a um ponto de inflexão, porém, devido à pequena amplitude entre os picos, não pode ser relacionado ao máximo e mínimo local. Esse problema pode ser resolvido por meio do uso da segunda derivada associada aos máximos e mínimos locais na região delimitada pelas bordas

Veja a Figura 3.22 onde dois gráficos sobrepostos, a primeira (em azul) e segunda (em vermelho) derivadas da mesma poça de soldagem da Figura 3.21. Note que no gráfico da primeira derivada no intervalo delimitado entre as bordas da poça é possível visualizar no gráfico da segunda derivada somente os pontos de máximo e mínimo relacionados a posição média do arame de soldagem. A posição do arame é determinada pela posição média entre os pontos de máximo e mínimo locais do gráfico da segunda derivada restritos a região entres bordas externas da poça.

O software de análise de imagens *offline* produzido em *C/C++* a partir do programa produzido em *MATLAB* deve filtrar a imagem e medir do tamanho da poça e posição do arame. Além disso, foram adicionadas mudanças ao programa como: a análise de uma série de arquivos já capturados em experimentos anteriores e a implementação de um filtro que detecta somente imagens que contenham somente a poça de soldagem sem brilho do arco.

A metodologia mais simples para detectar uma imagem que contenha somente a poça de soldagem sem brilho baseia-se no fato que nas imagens que não estejam exibindo a

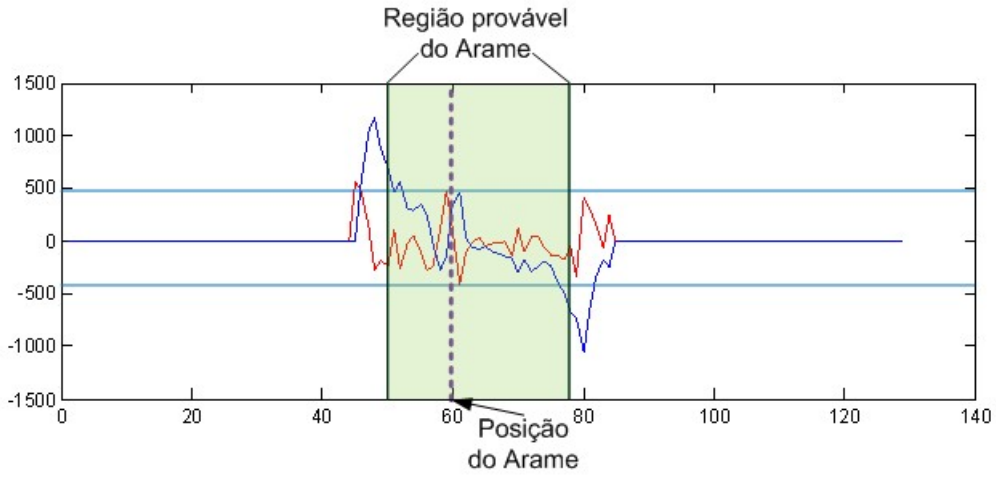

Figura 3.22: Identificação da posição do arame

poça possuam uma um brilho excessivo ou seja, o somatório da intensidade de brilho de todos os *pixels* da imagem é maior que nas imagens que exibam somente a poça sem brilho, porém isso só pode ser aplicado devido ao alto contraste entre a poça e o fundo que é preto puro (zero) se filtrado.

Por isso, o filtro para detectar imagens em curto-circuito que exibam somente a poça deve ser baseado em uma região provável de brilho máximo e mínimo obtidos a partir de experimentos em condições conhecidas no qual pode-se obter esses parâmetros. O problema é que para tipos diferentes de soldagem deve-se fazer novamente essa calibração a fim de obter novos valores máximos e mínimos compatíveis com esse experimento.

A vantagem desse programa de análise *offline* é a de permitir a determinação do brilho das poças obtidas em experimentos anteriores realizados em condições bem definidas de soldagem, para então determinar os valores de brilho máximo e mínimos de uma poça com brilho mínimo para experimentos posteriores.

3.5 EXPERIMENTO AVANÇADO

Os experimentos anteriores se mostram eficientes até certo ponto, pois problemas limitaram os resultados e a maioria deles poderiam ser solucionados resultando em uma melhora considerável do sistema de medição. Os problemas mais comuns eram a subtilização dos sistemas de captura, a dificuldade em se obter um imagem em foco, dificuldade em se determinar a abertura do diafragma, a falta de interação do software de captura, a dificuldade de calibrar o processo de filtragem de imagens sem brilho.

3.5.1 Subutilização do sistema de captura

O primeiro problema está relacionado à não utilização de todo potencial do sistema de captura de imagens que devido à distância focal estar próxima do 500 mm (objetiva de 25 mm) resultou em uma poça com o tamanho 30 *pixels* dos 1024 oferecidos pela câmera, além disso, a taxa de captura máxima obtida foi de 150 fps dos 100.000 fps definido no manual do fabricante da câmera.

Para resolver o problema da distância focal, introduziram-se anéis espaçadores entre a câmera e a objetiva de 25 mm. Por tentativa e erro, escolheram-se 2 anéis espaçadores que, quando montados, totalizavam um acréscimo de 15.9 mm entre o sensor **CMOS** e a objetiva. Isso resultou na redução da distância focal e transformou a câmera em um sistema *macro*. Após a colocação dos anéis a distância focal reduziu-se para 120 mm e resultou em um aumento tanto no tamanho quanto na quantidade de detalhes das imagens obtidas. A largura de um cordão de solda passou de 30 *pixels* para 300 *pixels* nas mesmas condições. Compare a Figura 3.23 e note a diferença entre poças obtidas em diferentes configurações de distância focal.

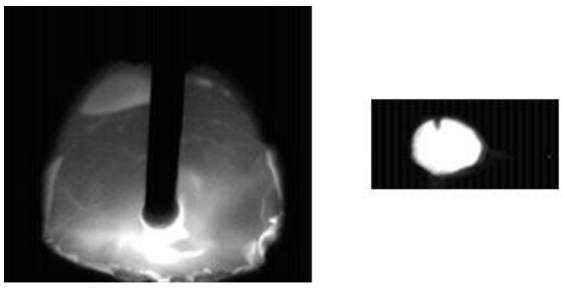

Figura 3.23: Poças em diferentes configurações de foco

Para resolver o problema da taxa de captura máxima foi necessário alterar **diretamente** as condições de **ROI** no programa *PFRemote* para então **somente** observar os resultados no programa *CamExpert* e alterar as configurações de câmera de acordo com os resultados a serem obtidos.

O programa *PFRemote* altera as características físicas da câmera, por isso é que se deve alterar nele antes de ser utilizado no programa *CamExpert* que é responsável somente pela captura da imagem obtida da câmera. Os passos exibidos na Figuras 3.24 e 3.25 permitem que o usuário defina a taxa máxima de captura suportada pelo *frame grabber*.

Segundo a Figura 3.24, primeiro deve-se abrir o programa *PFRemote* conectar-se à

câmera na aba *window* (A) e definir o tamanho da Janela (B). Nesse instante, toda vez **que se alterar um parâmetro, deve-se teclar enter, sob pena de o parâmetro não ser alterado**. Note que o *exposure time*, ajustado na aba *Exposure,trigger* e definido para uma taxa de amostragem em uma janela **ROI** de 1024 x 1024 *pixels*, é alterado automaticamente ao reduzir o tamanho da janela. Em razão disso a velocidade de amostragem aumenta (D).

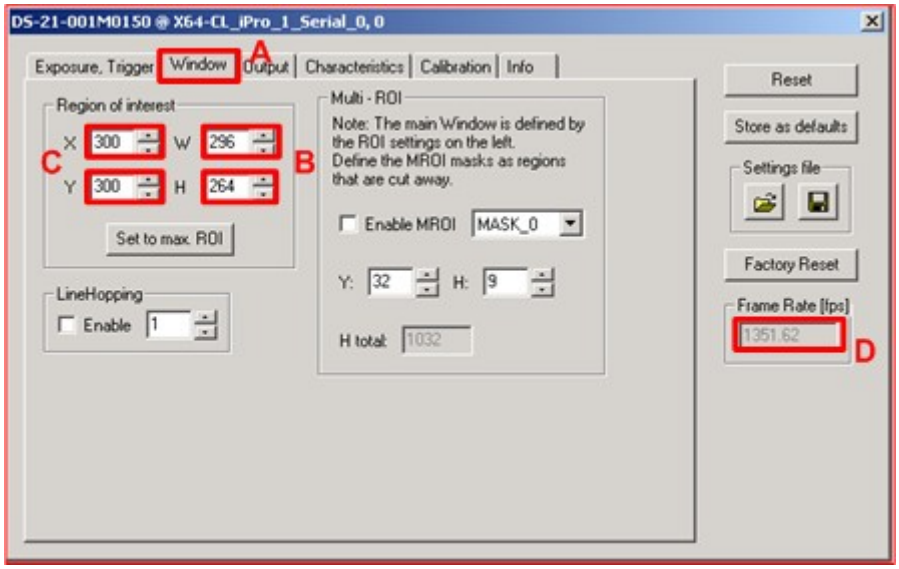

Figura 3.24: Definindo uma taxa de amostragem mais alta

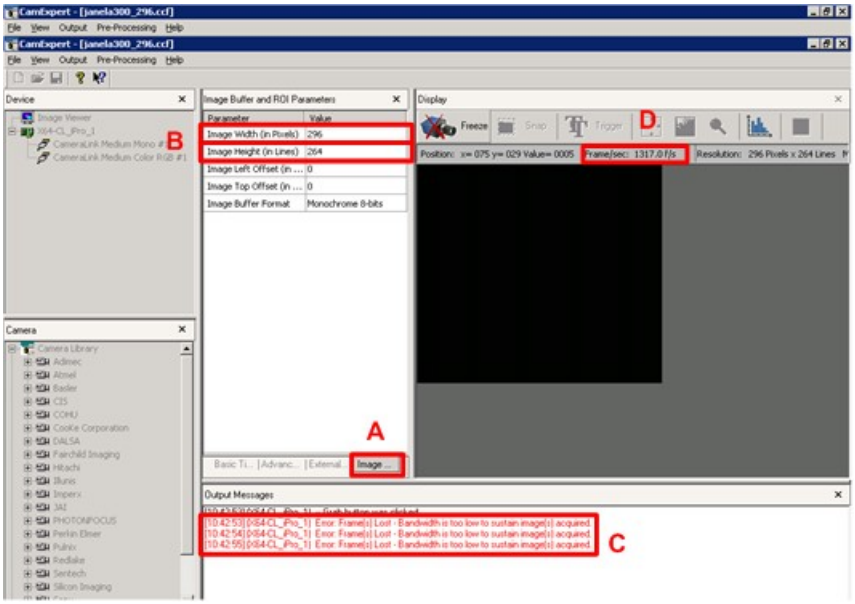

Figura 3.25: Definindo a maior taxa de amostragem suportada pelo *frame Grabber*

Para definir a velocidade máxima de captura segundo a Figura 3.25 deve-se abrir o programa *CamExpert* para configurar a câmera em estudo e então ajustar o tamanho da janela de captura exatamente como definida no programa *PFRemote* na aba *Image* (A), (B) e iniciar a captura.

Devem-se sobrepor as telas dos programas *PFRemote* e *CamExpert* de forma que seja possível observar o *log* do programa *CamExpert* e então deve-se reduzir sucessivamente o *exposure time* no programa *PfRemote* até que uma mensagem de erro seja observada (C). É nesse ponto que se encontra a maior taxa possível de captura pelo *frame grabber* para um quadro de determinado tamanho. Note que o valor definido em (D) pode não ser o valor máximo de captura de imagens. O verdadeiro valor é o exibido no programa *PfRemote*.

Os resultados dessas alterações permitem que a captura chegue a taxas elevadas somente limitadas pela capacidade de captura do *frame grabber*. A utilização dessa técnica permitiu que fosse possível obter quadros a uma taxa de 1351 fps para uma janela de 296 X 264 *pixels*.

3.5.2 Foco e Braço Robótico

Um dos problemas enfrentados em todos os experimentos foi a necessidade de se definir o foco e a abertura do diafragma próximo da tocha de soldagem com um filtro óptico já acoplado à câmera. Devido à câmera utilizada nesse trabalho requerer muita luz para se obter uma imagem nessa condição, foi necessário desenvolver uma estratégia para definir um foco e uma abertura de diafragma, já que o foco é alterado quando filtros ópticos são acoplados (o ângulo de refração varia com a freqüência do feixe luminoso incidente nas lentes).

A solução encontrada foi utilizar um holofote de 500W próximo à tocha para que a energia luminosa na região do infra-vermelho próximo seja capturada pela câmera. Essa tática, exibida na Figura 3.26, funcionou bem para o filtro infravermelho, bastando colocar um cordão de solda sólido abaixo do bocal da tocha de soldagem, colocar o holofote apontado para o cordão e em direção à câmera e reduzir o tempo de exposição do programa *PFRemote* para algo muito próximo de 10 fps, para então, observar os resultados no programa *CamExpert* e alterar o foco até encontrar a qualidade desejada.

O grande problema do ajuste manual do diafragma é que este é empírico. Isso implica que o operador deve realizar alguns experimentos até obter a habilidade para definir o ajuste. Isso pode comprometer a qualidade de medição, uma vez que se trata de um parâmetro definido manualmente, logo sensível a avaliações subjetivas.

Outro problema de implementação ocorrido foi a escolha de como deveria ser acoplado o conjunto da câmera: primeiramente, utilizou-se um tripé, depois, um braço robótico

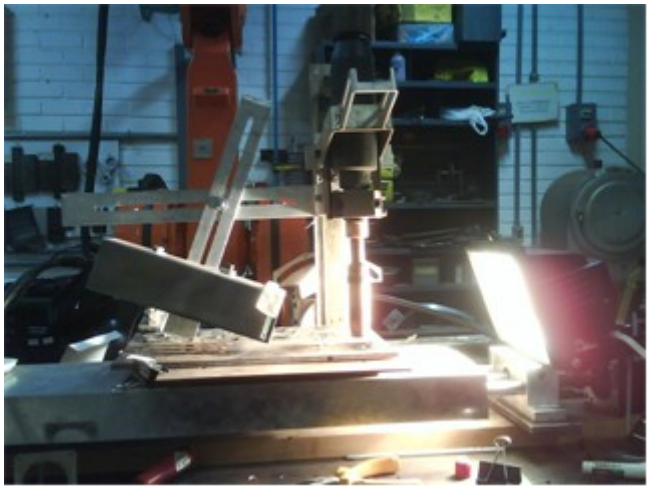

Figura 3.26: Montagem para obter o foco da poça

e, por último, foi adaptado um suporte rígido à tocha de soldagem (Figura 3.26).

3.5.3 Programa de aquisição e falta de interatividade

O programa de captura e processamento de imagens foi todo desenvolvido em C e para que houvesse algum ajuste era sempre necessário recompilá-lo. Durante o processo de captura somente era visível uma tela preta sem interação com o usuário.

Para resolver esse problema foram implementados menus em modo texto para que um usuário com pouca experiência conseguisse realizar o básico, ou seja, capturar e ou processar imagens por um período determinado pelo próprio usuário.

Foram desenvolvidos 2 menus o primeiro para que o usuário escolhesse o tipo de experimento como: (a) captura somente video; (b) captura video e imagens; (c) captura processa e armazena video e imagens. O segundo menu consiste em configurar o tempo de captura e os arquivo de configuração de câmera.

3.5.4 Determinação de um cordão de soldagem padrão

Um dos primeiros experimentos propostos foi definir um cordão de solda padrão pois, em experimentos anteriores, os cordões eram variados a cada experimento. Isso acarretava problemas de instabilidade do arco, produzindo efeitos tais como: a ocorrência de formação de modos de transferência globular e a formação de porosidade.Por isso foram realizados vários experimentos para se obter o melhor cordão em função de conjunto de características, a fim de formar um cordão de qualidade adequada, com um processo estável no modo curto circuito.

O primeiro problema a ser resolvido foi a colocação de um novo bocal do arame de soldagem, já que o anterior estava muito curto propiciando a formação de um arco instável, além disso, foi determinado um *standoff* padrão de 16 mm (resultando em uma distância do tubo de contato à base de 20 mm) para a tocha de soldagem em todos os experimentos.

Definidos os parâmetros físicos da tocha de soldagem, foram então fixados uma série de parâmetros para então variar somente a velocidade de alimentação do arame. O primeiro ajuste a ser fixado foi a vazão do gás de proteção que era da ordem de 12 l/min e foi elevado para um valor de 16 l/min. O segundo parâmetro fixado foi a velocidade da mesa linear, que em experimentos anteriores era de 4 mm/s e foi elevada para 5mm/s. O terceiro parâmetro a ser fixado foi a tensão do arco de soldagem, que anteriormente variou de 20 a 23 V e foi fixada em 20 V para evitar instabilidades anteriormente observadas.

Os melhores cordões de solda foram obtidos com velocidades de alimentação do arame em torno de 6 m/min. Esses cordões são bem definidos, densos e sem falhas. Esses parâmetros foram utilizados em todos os experimentos restantes a fim de garantir uma espessura do cordão conhecida além de facilitar o ajuste de alguns parâmetros de configuração da câmera como o **ROI**. A Figura 3.27 Mostra alguns exemplos desses cordões.

Figura 3.27: Fotos de Cordões Padrões

3.5.5 Determinação de uma inclinação da câmera

É determinante em um bom experimento de captura de imagens a definição da posição da câmera em relação ao objeto em estudo, nesse caso a poça fundida. Por isso, inicialmente o objetivo era obter uma poça em perfil, para então migrar para uma posição superior frontal, como mostrado na Figura 3.23.

Para a obtenção desse tipo de imagem, adaptou-se um suporte de câmera acoplado diretamente na tocha de soldagem com a caixa de proteção inclinada em torno de

30 ^{o} em relação à superfície de soldagem, conforme se pode observar na Figura 3.28. Um do problema relacionado a este suporte adaptado é que o ângulo pode variar com o tempo. Isso é causado, principalmente, pelo fato de o conjunto óptico (caixa de proteção, câmera, filtro) estar fixado por meio de um único parafuso, o que pode propiciar a rotação em torno de seu eixo. O uso da técnica do holofote permite a detecção deste problema no período de ajuste anterior ao experimento.

Figura 3.28: Montagem do suporte e a possível falha no posicionamento

3.5.6 Proposição e execução de experimentos

A partir da montagem definitiva do sistema de captura de imagens com um foco restrito a 12 cm, diafragma ajustado, suporte ajustado em um determinado ângulo próximo aos 30 ^{o} e filtro óptico infravermelho acoplado à câmera, foram propostos vários experimentos, visando à ajustar o conjunto de componentes do sistema (*hardware* e *software*), a fim de obter uma configuração que propiciasse a melhor imagem da poça de fusão.

O primeiro experimento proposto foi a captura a uma taxa de 1000 fps, utilizando o programa de captura já utilizado nos experimentos anteriores com pequenas mudanças, como o aumento da quantidade de quadros alocados na memória e a redução do tempo de captura. Essas pequenas mudanças foram necessárias devido à alta taxa de amostragem, que, para um experimento de 10 segundos, produzia até 10000 imagens, o que facilmente estouraria o *buffer* de memória. Por isso o experimento limitou-se a apenas 5,5 segundos de captura.

Os resultados desse experimento foram satisfatórios, pois foi possível observar muitos detalhes ainda não observados como: a mudança de cada quadro, diversos quadros totalmente sem arco, a propagação de uma frente de onda causada pela explosão do arame com a poça e a variação da posição do arame de soldagem. Observou-se também

a formação e movimentação de uma mancha de coloração uniforme sobre a poça, o que acredita-se ser escória.

A Figura 3.29 mostra um conjunto de imagens em que é possível obter uma imagem totalmente sem arco (linha 1 coluna 2 e linha 2 coluna 1) e sua seqüência (esquerda para a direita), a propagação da frente de onda e sua seqüência (linha 1 colunas 8,9 e linha 2 coluna 6,7), o acúmulo de escória sobre a poça em todas as imagens ao fundo e nas bordas da poça. Note que essas 18 imagens foram retiradas de um período de 5,5 segundos que contêm 5488 imagens.

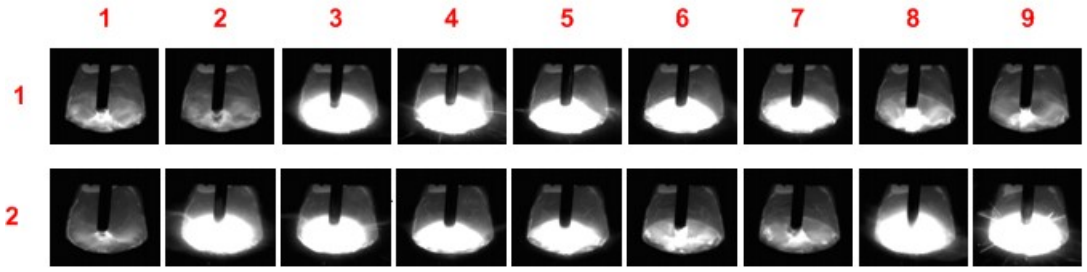

Figura 3.29: Seqüência de quadros em uma captura com taxa de 1000 fps

Apesar da evolução desse experimento em relação aos anteriores alguns problemas ocorreram como a sobreposição de 2 quadros em uma mesma figura, a grande quantidade de arquivo em disco gerado para um curto experimento de 5,5 segundos (praticamente 900 MB) além disso os *players* de video possuem dificuldade em executar um arquivo de video de 1000 fps.

Um provérbio já conhecido diz que uma imagem vale mais que mil palavras em nosso caso como o papel não permite anexar vídeos e todas as imagens capturadas de um experimento, por isso todos esses experimentos podem ser encontrados em um **DVD** anexo a cada cópia desse trabalho.

O experimento realizado é ideal para análise de transferência metálica, mas não para medição da poça de soldagem, pois não são necessários todos esses quadros para analisar, mas somente os quadros em curto-circuito. Por isso o segundo experimento proposto foi de utilizar o detector de curto-circuitos para sincronizar e restringir a captura de imagens a somente imagens sem brilho.

Para esse experimento foi montado o sistema de medição de tensão do arco conectado ao sistema de detecção de curto-circuito e um sistema de aquisição de dados, além

disso, foi necessário configurar o arquivo de captura da câmera para adquirir imagens somente se um pulso de 80 us fosse injetado na porta do *trigger* 2.

No período de realização do segundo experimento, foi montada a segunda placa mãe do computador industrial que tem o intuito de abrigar a placa de aquisição de dados **EAGLE PCI-703S-16A** para realizar a aquisição da tensão, corrente, amplitude sonora, de todos os experimentos de soldagem do laboratório de soldagem. Para isso um programa de aquisição foi desenvolvido em conjunto com os alunos do laboratório de soldagem em *LabView* apara atingir esse fim.

Para realizar o experimento foi primeiro gravado um código fonte com o programa de janelamento no detector de curto-circuito a fim gerar de forma automática a definição de limiares para disparo do *trigger*.

Os experimentos foram realizados com as mesmas configurações do experimento de captura livre, além disso, os dados da tensão, pulso do *trigger* foram capturados pela outra placa mãe do computador industrial por um software independente de captura desenvolvido em *LabView*.

Os resultados desses experimentos foram muito satisfatórios, pois foi possível reduzir de forma considerável o número de arquivos capturados para um experimento de 5 segundos, foram coletados 483 imagens ou seja possivelmente 483 curtos-circuitos. A Figura 3.30 mostra uma seqüência de imagens; note como não há imagens com brilho excessivo, o que mostra que o filtro adaptativo funcionou conforme esperado.

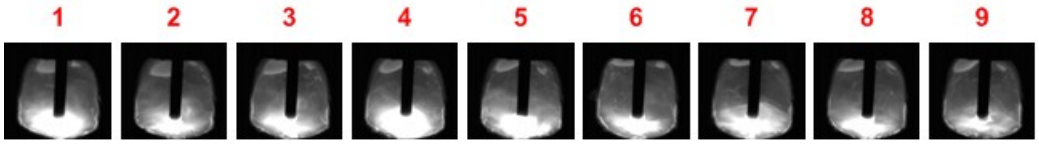

Figura 3.30: Seqüência de captura sincronizada com uso de filtro adaptativo

Para verificar se o detector de curto-circuito estava funcionando de forma correta, foram capturados simultâneamente os sinais de *trigger* e de tensão do arco. A Figura 3.31 mostra o resultado de uma janela. Note que em grande parte é possível observar o sincronismo entre os curtos-circuitos e o sinal de *trigger*, porém em (A) houve uma falha em detectar o curto e em (B) houve um falso positivo.

Note na Figura 3.30 que ainda é possível observar a presença de arco voltaico residual em grande parte dos experimentos e como nos experimentos de captura livre foi possível

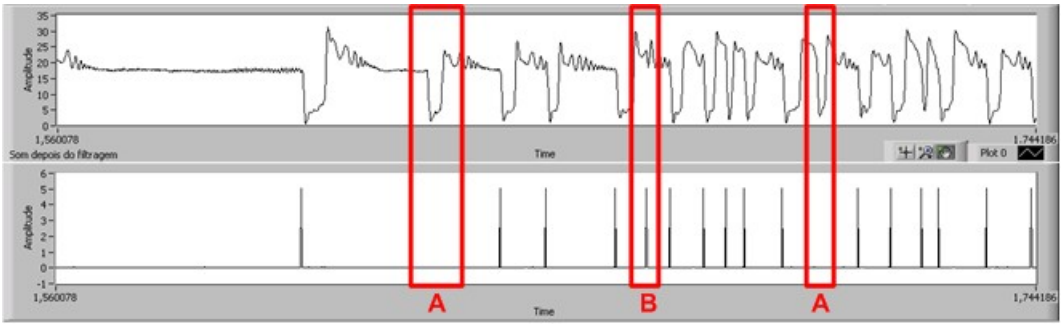

Figura 3.31: Comparação dos sinais de tensão do arco e o *trigger* de filtro adaptativo

obter imagens com nenhum arco, optou-se por propor uma série de experimentos a fim de determinar de forma manual os limiares de tensão máximos e mínimos a fim de melhorar ainda mais as imagens obtidas.

Essa série de experimentos utilizou como parâmetros os resultados apresentados no programa de captura dos sinais de tensão e *trigger*, aliado às imagens capturadas para determinar o menor e maior valor de *threshold* comparado com o tipo de imagem obtida.

O valores mínimo e máximo de *threshold* adotados inicialmente foram 70 e 400 **AD** respectivamente, ou seja, para um conversor **AD** de 10 bits com a tensão de comparador fixada em 5 V, uma tensão de 5 V é proporcional a 1024 **AD**, logo 70 **AD** é 0,34 V e 400 **AD** é 1,95 V. Como o medidor de tensão do arco existente possui uma redução aproximada de 10:1, logo a tensão do arco mínima é 3,4 V e máxima 19,5V.

Para realização do ajuste manual foi proposto primeiramente reduzir sucessivamente o limiar mínimo a fim de forçar que os quadros fossem obtidos no ponto de mínimo, até que o detector de curto-circuitos falhasse em sua tarefa, Complementar a isso, foi também reduzido o limiar máximo para que evitar a ocorrência de falhas como na Figura 3.31 (B).

Os resultados desses experimentos foram satisfatórios, porém com um custo de tentativas elevado a fim de adaptar a captura a um único tipo de cordão em condições específicas de soldagem. Por isso **para generalizar o sistema de medição, deve-se utilizar sempre o sistema de detecção de curtos com janelamento**.

A Figura 3.32 mostra uma seqüência de imagens em que foi possível reduzir ainda mais o arco residual das imagens obtidas.

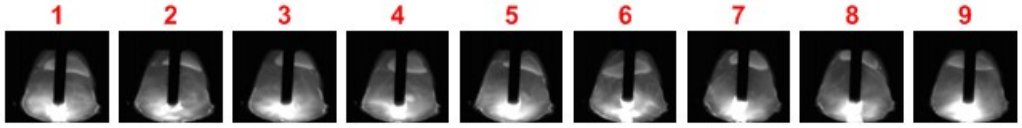

Figura 3.32: Seqüência de imagens com limiares determinados experimentalmente

3.5.7 Proposição um novo algoritmo de medição

A obtenção de novas imagens bem diferentes das anteriores conduziu a um novo algoritmo de medição que adotou alguns fundamentos já testados com sucesso em 3.4.2. Além disso, as novas imagens propiciaram definir novos parâmetros de soldagem além da posição do arame e da largura da poça, como a penetração do arame na poça.

Devido às diferenças entre as imagens dos experimentos anteriores foi proposto alterar como deve ser realizada a medição da imagem. A primeira atitude foi não mais aplicar a somatória das linhas verticais em todo o conjunto da figura, mas sim em regiões. Isso permite melhorar a exatidão da medida. Outra grande vantagem é eliminar o uso de filtros lógicos para retirar imagens com brilho excessivo e o filtro de *threshold* para aguçar as bordas da poça. Isso implica em maior eficiência do algoritmo de aferição.

Assim como foi realizado anteriormente foi desenvolvido um código fonte do algoritmo para uma análise offline em *MATLAB*, para então portá-lo para C e finalmente integrá-lo ao programa de captura de imagens.

O algoritmo deve funcionar da seguinte forma:

1. Retirar *Scanlines* da imagem original por meio de um filtro *threshold* para níveis de cinza menores que 6
2. Aplicar um filtro tipo mediana com uma janela de tamanho 3X3
3. Definir uma linha inicial onde existe a possibilidade de se encontrar o arame (a linha central da imagem, por exemplo)
4. Somar as colunas em uma região vertical definida pela linha inicial e gravar em um vetor
5. Calcular a primeira derivada sobre o vetor e associar os mínimos e máximos locais com as bordas do arame

6. A partir das bordas do arame encontrar a zona central do arame
7. Calcular a primeira derivada sobre a linha vertical definida pela zona central do arame e gravar em um vetor
8. Associar o máximo local do vetor à posição de toque do arame
9. Somar as colunas em uma região vertical definida pela linha de toque do arame e gravar em um vetor
10. Calcular a primeira derivada sobre o vetor e associar os mínimos e máximos locais com as bordas da poça fundida ignorando a parte central onde se encontram as bordas do arame de soldagem

Para elucidar o algoritmo, nada melhor que imagens, para isso, cada passo será representado por um conjunto de imagens para permitir a visualização da evolução do algoritmo. Os itens 1, 2, 3 e 4 do algoritmo podem ser observados na Figura 3.33 em (A),(B),(C) e (D) respectivamente.

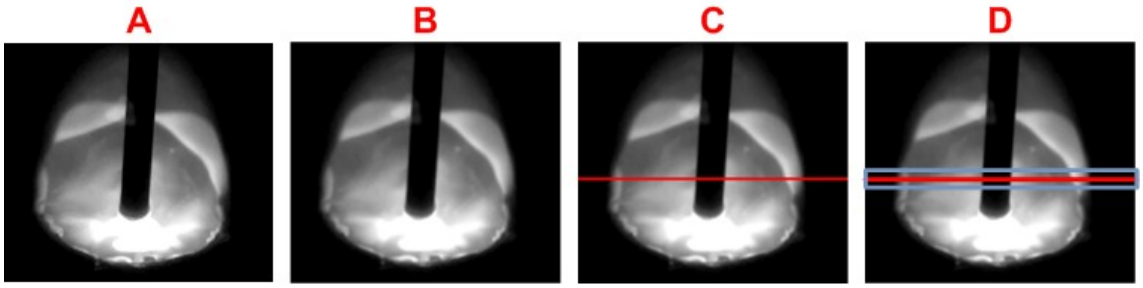

Figura 3.33: Passos 1 (A),2 (B), 3 (C) e 4 (D)do algoritmo de medição

No passo 2, o objetivo de aplicar o filtro de mediana é suavizar o contorno das bordas e retirar ruídos da imagem. No passo 3 deve-se arbitrar uma linha onde seja possível encontrar o arame para então no passo 4 definir uma área em torno dessa linha e então somar as colunas dessa região para formar um vetor.

No passo 5, esse vetor deve ser diferenciado para, a partir dos pontos de máximo e de mínimo, obter as bordas do arame de soldagem. Note que na Figura 3.34 (B) é possível diferenciar os picos de máximo e mínimos das bordas do arame das bordas da poça de soldagem fundida devido a brusca variação da intensidade de *pixels* na transição entre a poça fundida, ao fundo, com arame, sobreposto.

No passo 6, as posições das bordas do arame permitem definir uma posição central, que é utilizada como referência no passo 7. Este diferencia uma linha vertical que cruza o

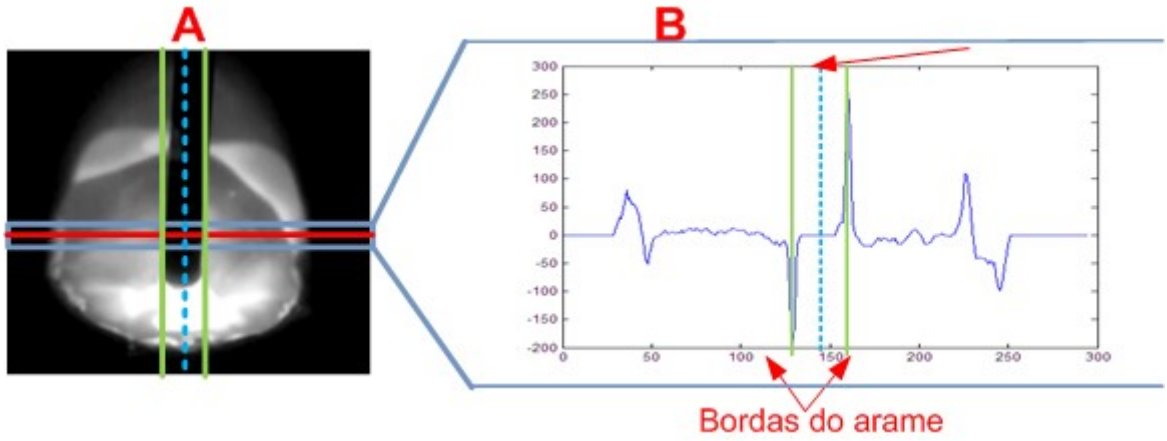

Figura 3.34: Passos 4,5 e 6 do algoritmo de medição

ponto central para relacionar o ponto de máximo ao provável ponto de toque do arame sobre a poça. Essa detecção é realizada no passo 8. O resultado da aplicação desses 3 passos pode ser vista na Figura 3.35)

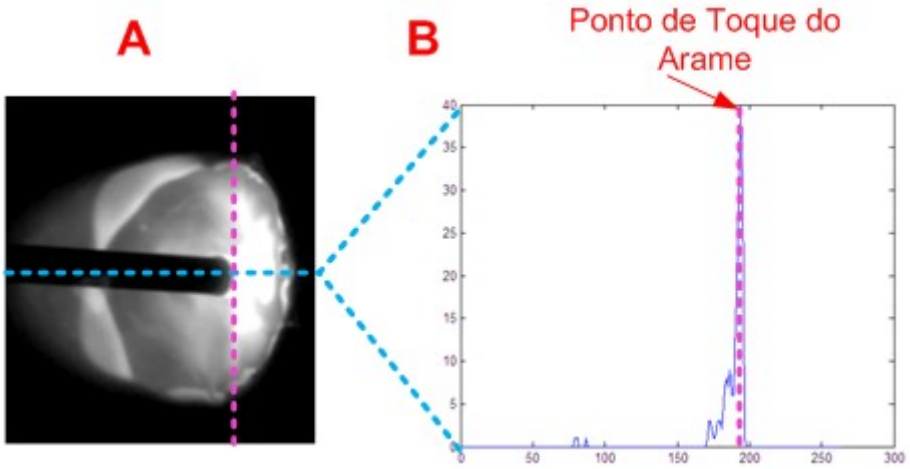

Figura 3.35: Passos 6,7 e 8 do algoritmo de medição

O ponto de toque é o local onde se pode encontrar a poça com a menor distorção de perspectiva em relação ao arame. Por isso, **nesse ponto** é possível relacionar diretamente, com uma boa precisão, o tamanho da poça com o diâmetro conhecido do arame (1 mm).

Finalmente, no passo 9, deve-se aplicar novamente uma soma das colunas verticais na região definida pelo ponto de toque para então aplicar a derivada sobre esse vetor como pode ser visto na Figura 3.36.

No passo 10, também exemplificado na Figura 3.36, como se desejam obter as medidas

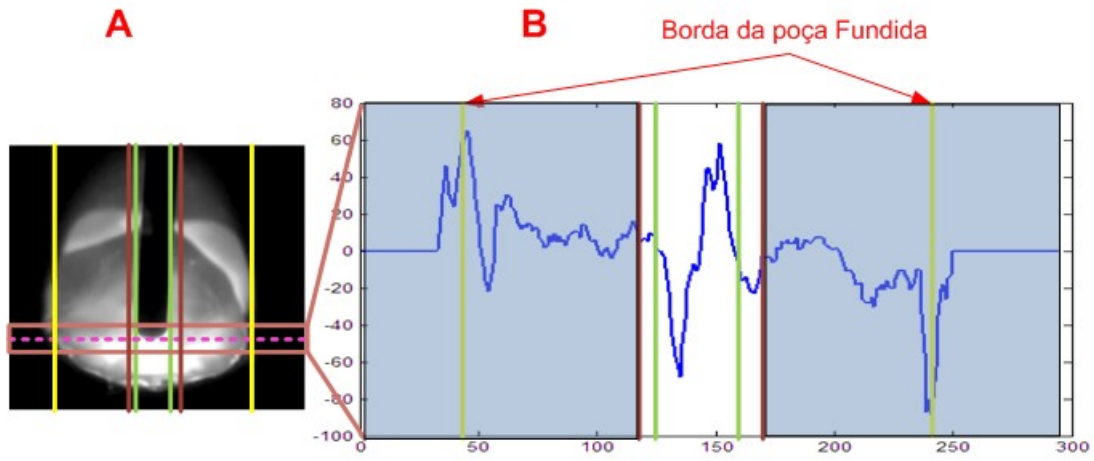

Figura 3.36: Passos 9 e 10 do algoritmo de medição

da borda da poça e sabe-se que as bordas da poça não se encontram na região central, esta foi ignorada antes de se associarem os mínimos e máximos. Por esse motivo, houve a necessidade de se obter a posição do arame nos passos anteriores.

As novas versões dos programa de medição da poça, tanto em *MATLAB* quanto em C, são explicitadas nos Anexos E e G.

Uma novidade foi adicionada no programa de medição *offline*: a colocação de linhas suporte indicando onde o algoritmo definiu: (a) a posição das bordas do arame e da poça; (b) indicar o ponto de toque do arame. A Figura 3.37 mostra diferentes tipos de medições para cada tipo de imagem note que o programa de medição ainda pode cometer erros em (C) e(D).

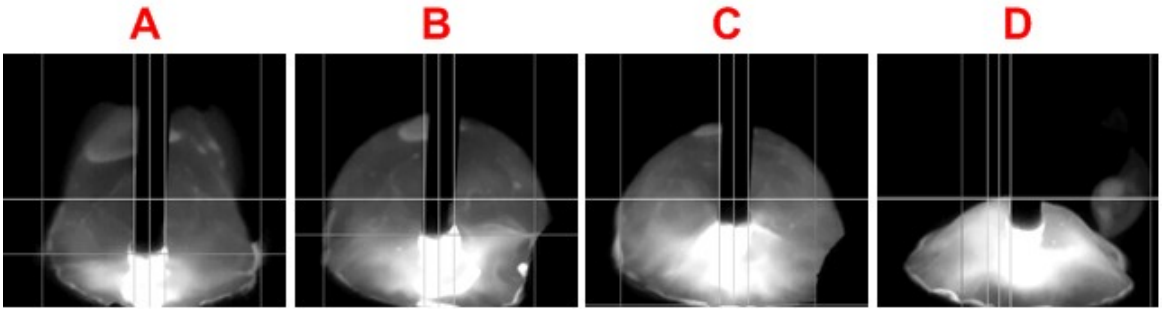

Figura 3.37: Exemplos de imagens analisadas pelo programa *offline*

3.5.8 Experimentos Finais

O novo algoritmo de medição após ser integrado ao sistema de captura utilizado em todos os experimentos anteriores, completou todas a necessidades ainda pendentes para

realização de um experimento completo, ou seja, capturar imagens da poça fundida somente nos períodos de curto-circuito para então medí-las.

Com as mesmas configurações de soldagem definidas em 3.5.4 foi proposto realizar um experimento de captura e medição a fim de verificar a eficiência do algoritmo.

O algoritmo de captura e medição deve: (a) armazenar em memória **RAM** um quadro que foi obtido a partir do disparo do *trigger* do detector de curto-circuitos; (b) processá-lo a fim de obter a posição das bordas do arame e poça fundida; (c) aferir a posição de toque do arame do soldagem na poça; (d) armazenar o tempo entre os quadros. Todos esses dados devem armazenados em **RAM**, caso algum programa queira acessar esses dados em tempo real. Somente ao final da captura esses dados serão armazenados em disco.

Os resultados assim como aos anteriores foram muito satisfatórios as imagens obtidas são de boa qualidade como na Figura 3.38. Na figura é possível observar 3 tipos imagens de captura: em modo livre de captura sem medição; imagens sincronizadas com valores de *threshold* forçados; com técnica de janelamento. O mais interessante são os resultados das medidas. Todas as medidas realizadas foram relacionadas em *pixels* em relação ao eixo vertical.

A Figura 3.39 mostra o resultado da medição de uma série de 214 quadros sucessivos, apresentados em forma de gráfico, o qual tem a finalidade de mostrar o perfil do cordão fundido. É possível observar o caminho das bordas laterais da poça de soldagem fundida e o caminho das bordas do arame ao centro. Note que o algoritmo não é eficiente em medir o inicio da formação do cordão, como mostra a Figura 3.37 (D).

Note que na Figura 3.39 é possível observar as variações do arame e como a poça não é muito afetada por essas pequenas variações, provavelmente devido à alta tensão superficial da poça fundida.

Ao aplicar as imagens obtidas dos experimentos da Figura 3.38 no algoritmos de análise *offline*, percebe-se que a medição tende a ser mais ruidosa na medida que a figura se torna mais escura ou fora de foco como apresentado na Figura 3.40.

Uma medida interessante é a penetração do arame de soldagem na poça fundida apresentada na Figura 3.41 note que é possível observar uma tendência que com o passar

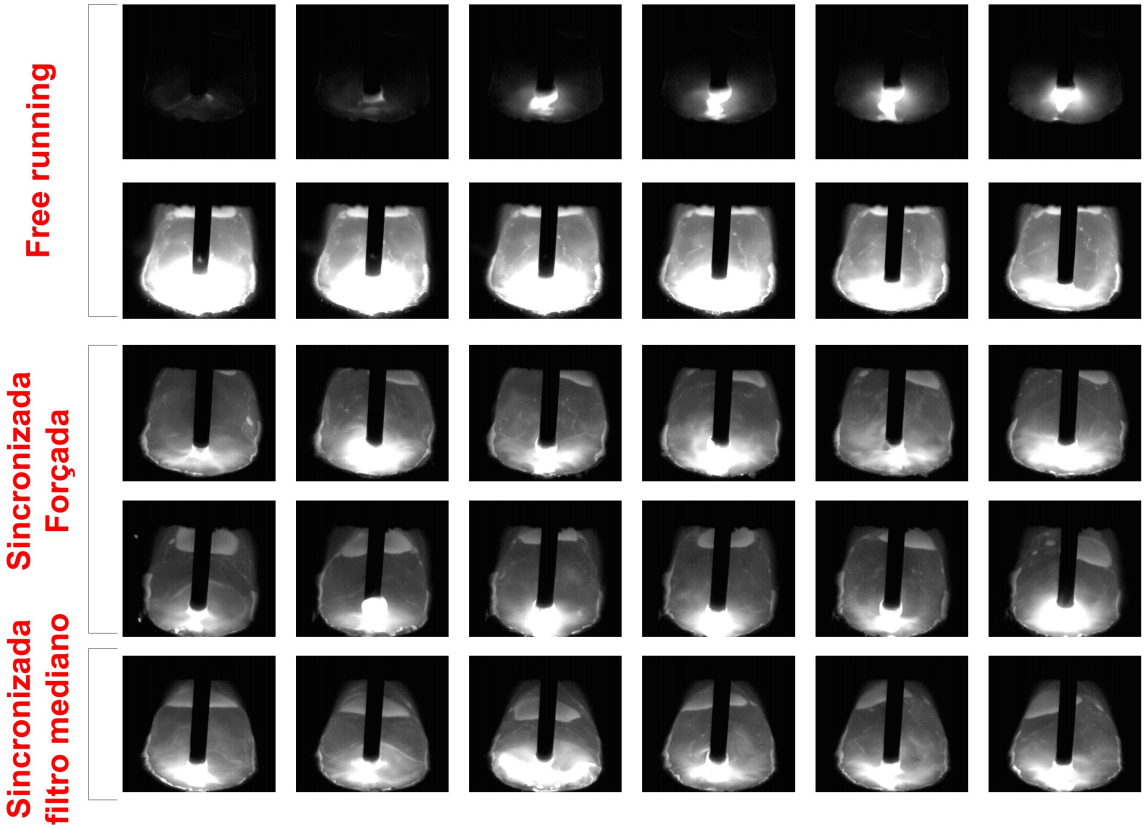

Figura 3.38: Exemplo de vários experimentos

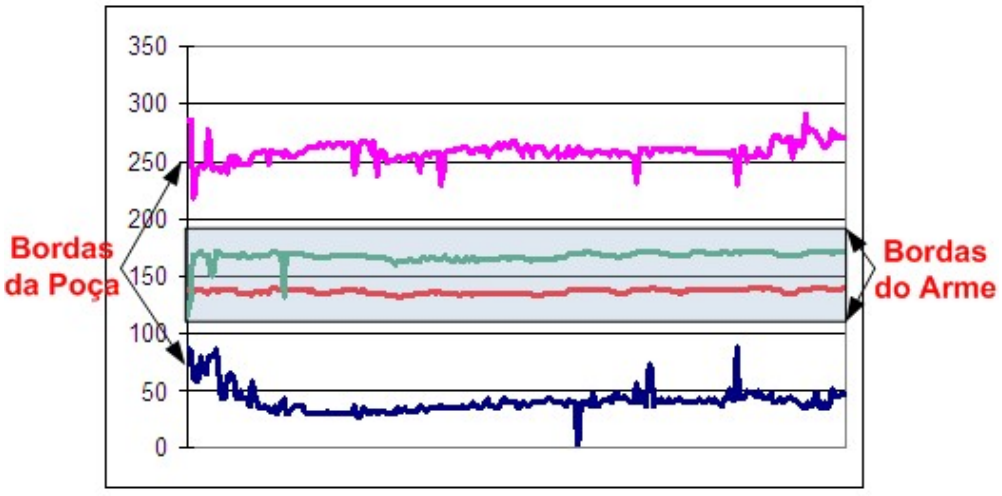

Figura 3.39: Perfil da medição de um cordão de soldagem (Posição em *pixels* por tempo)

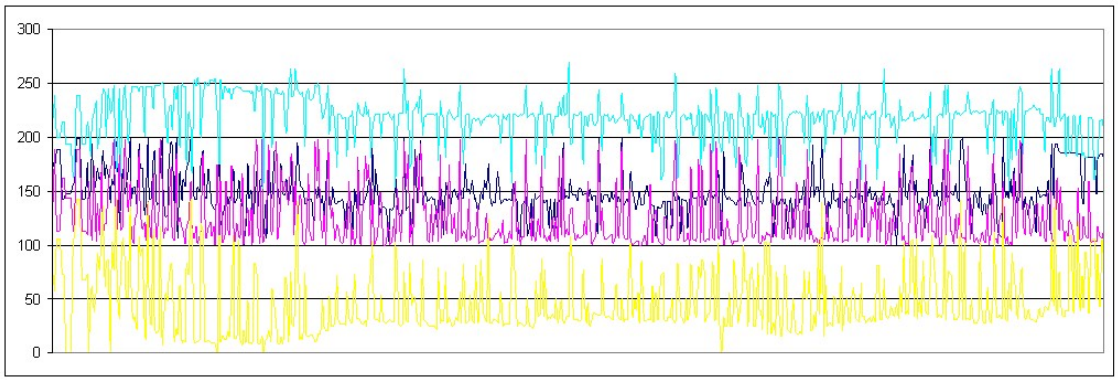

Figura 3.40: Gráfico de medição muito ruidoso (Posição em *pixels* por tempo)

do tempo o arame de soldagem toque a poça em uma altura cada vez menor em relação ao corpo de prova.

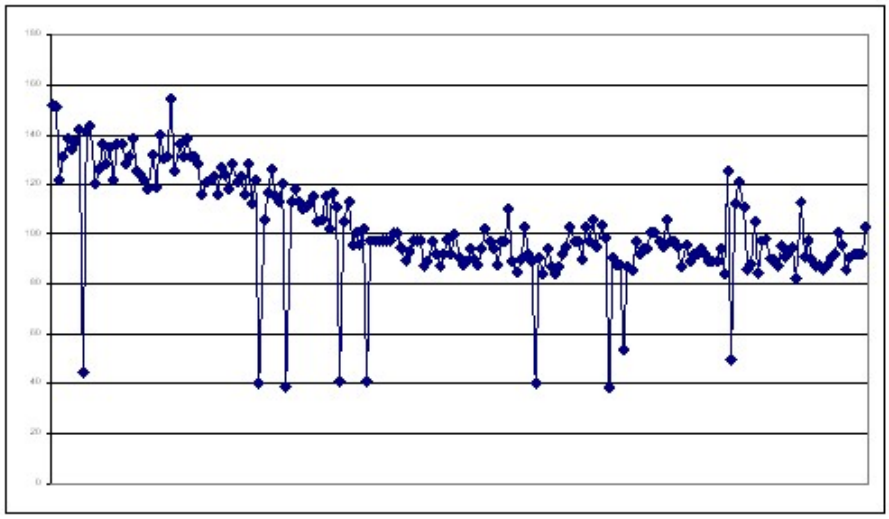

Figura 3.41: Gráfico da medição da altura da posição de toque do arame de soldagem no tempo (Posição em *pixels* por tempo)

Para verificar a eficiência do sistema de captura e medição um sistema de monitoração capturou os dados de tensão, e sincronização do detector de curto-circuitos. Um problema em usar somente esses parâmetros é que não há confirmação de que o quadro realmente foi capturado com o curto detectado. Por isso foram alterados alguns parâmetros de configuração da câmera para que o *frame grabber* respondesse de forma síncrona ao detector de curtos. Um dos parâmetros alterados foi o *Strobe Method Setting*, cujo valor deve ser ajustado *Method1*. Para que isso possa ser feito, é necessário executar o programa *CamExpert* e selecionar a aba *Advanced*, em que se encontra o campo relacionado a esse parâmetro (Vide Figura 3.42 (A)). Para concluir a sincronização, deve-se definir o tamanho do pulso que será retornado no campo *Strobe Duration* na Figura 3.42 (B).

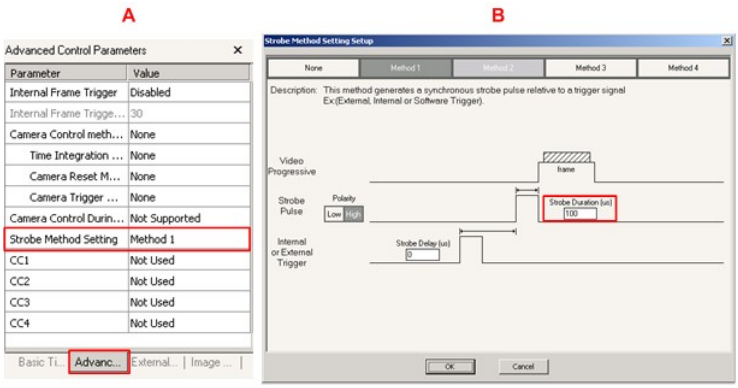

Figura 3.42: Configuração do pulso de obtenção de imagem

O último experimento apresentado nesse trabalho consistiu em comparar um gráfico obtido a partir da medição das poças fundidas a uma imagem do cordão de soldagem resultante, a fim confrontar o formato do gráfico com o formato do cordão resfriado.

Sabe-se que o algoritmo não foi eficiente em medir a poça no início de sua formação, por isso, ao realizar esse experimento, foi retardado o inicio da captura de imagens a fim de evitar a região de inicio da formação da poça fundida. O problema em utilizar essa técnica é a de não definir uma posição de referência entre o início do gráfico e o cordão de soldagem.

Os experimentos foram realizados com as mesmas configurações definidas em 3.5.4. Os resultados são apresentados de forma qualitativa, já que foi necessário haver uma transformação de escalas entre o gráfico obtido e a imagem fotografada do cordão de solda resfriado.

Os resultados são interessantes, pois é possível comparar o cordão solidificado ao gráfico resultante da medição com uma boa aproximação como observado na Figura 3.43. Note como é possível relacionar a posição inicial de medição ao início do cordão, bastando para isso comparar a inclinação das curvas mostradas na Figura com os limites laterais do cordão de solda.

Um detalhe que chamou atenção nesse experimento foi a observação de um desalinhamento momentâneo do arame, provavelmente provocado por sua curvatura excessiva decorrente do embobinamento. O sistema de medição foi capaz de detectar esse desvio, apesar de o mesmo não acarretar nenhum defeito expressivo visível a olho nú. (vide Figura 3.44)

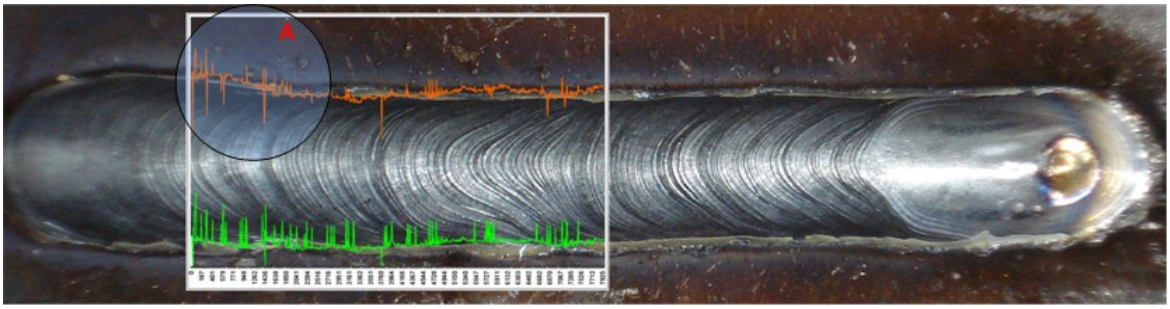

Figura 3.43: Montagem entre o cordão solidificado e o gráfico de medição da poça de soldagem

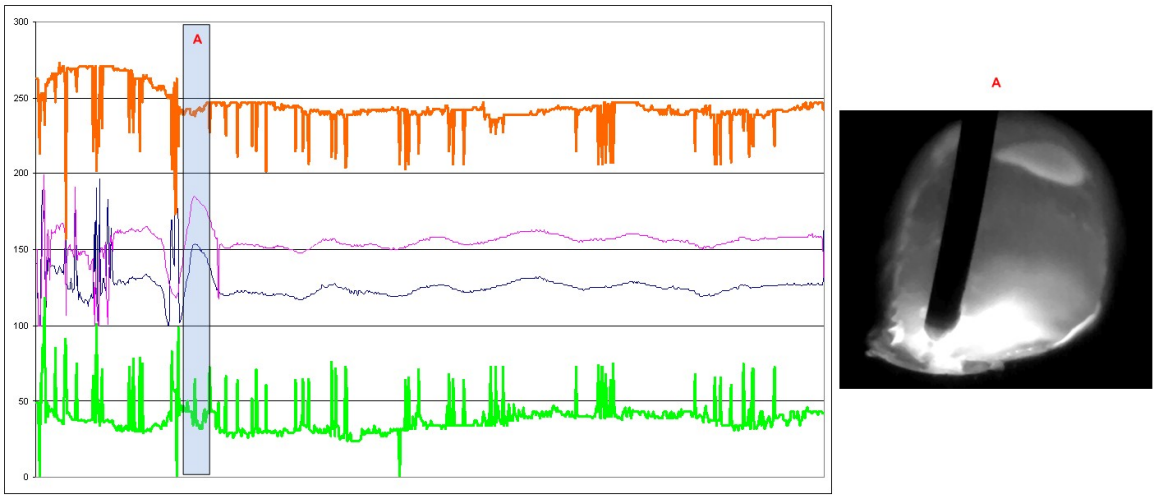

Figura 3.44: Detecção de uma falha no posicionamento do Arame de soldagem

Uma das grandes vantagens do sistema de medição da poça de soldagem é que se pode exportar esse sistema para um ambiente industrial que possibilitará formação de um registro de todo histórico de soldagem e utilização dos parâmetros geométricos da poça e de alinhamento do arame para realimentação em um sistema de controle de largura de cordão e seguimento de junta (*seam tracking*).

4 DISCUSSÃO DOS RESULTADOS

Este trabalho propôs a criação de um sistema de monitoramento do processo de soldagem **GMAW** por curto-circuito, baseado na aquisição de sinais elétricos do processo e de imagens da poça de fusão. A aquisição destas foi sincronizada aos períodos de menor luminosidade do arco, correspondentes aos períodos de curto-circuito. A sincronização permitiu a aquisição de imagens com distribuições de intensidade de brilho semelhantes, possibilitando o desenvolvimento de um algoritmo de processamento e medição de características geométricas da poça. Os parágrafos abaixo apresentam uma discussão aprofundada sobre o trabalho desenvolvido, assim como discorrem a respeito dos resultados obtidos e sugerem possíveis melhorias a serem implementadas no sistema.

O primeiro passo para consecução dos objetivos propostos envolveu a montagem de um aparato experimental complexo, o qual era composto de: (a) um equipamento computacional industrial de captura de imagens; (b) uma mesa de movimentação linear e respectivo sistema de controle; (c) um sistema eletrônico de detecção de curto-circuito e geração de pulsos de sincronização (*trigger*) e (d) um conjunto de software para captura e análise de imagens. A integração de todos esses elementos constituiu-se em um trabalho penoso, devido à grande quantidade de elementos necessários para serem ajustados e para trabalharem simultâneamente, de modo a realizar simples experimentos de soldagem. Os diversos elementos diferentes e dispersos tornam-se potenciais pontos de falha: um simples cabo mal conectado pode causar uma falha difícil de ser rastreada. Para facilitar a reprodução dos experimentos apresentados neste trabalho, descrevem-se no anexo A os passos necessários para montagem e execução de um experimento utilizando esse sistema. Outro anexo importante, B, descreve a montagem do equipamento computacional industrial de captura de imagens, em cuja implementação dedicou-se um tempo elevado. Esse anexo apresenta como montar esse tipo de equipamento, as vantagens técnicas, os principais componestes envolvidos e a instalação física. Além disso, mostra procedimentos para uma correta instalação do sistema operacional e *drivers* e para seu uso adequado.

Após a realização de testes preliminares para verificação do correto funcionamento do aparato experimental, realizou-se pesquisa sobre técnicas de monitoração e de análise

dos parâmetros de soldagem. Dentre os assuntos pesquisados, enfatizaram-se aqueles relacionados a condicionamento de sinal, a taxa de amostragem, a técnicas de monitoramento do processo, como a técnica de janelamento e extração de características (*features*) dos sinais de tensão e de corrente de soldagem ((Carvalho 1997)). Com base nesse estudo, desenvolveu-se um sistema de análise de sinais de tensão do processo e de detecção dos momentos correspondentes aos períodos de curto-circuito. A detecção dos curtos-circuitos baseou-se na utilização da **tensão média baixa** e da **tensão média alta** (vide seção 2.3), referentes aos dados capturados em uma janela anterior, para definir os limiares indicativos da presença de curto-circuito. Essa abordagem resultou em um filtro de detecção de curtos-circuitos que se adapta a variações possíveis no processo de soldagem **GMAW**, implicando em maior probabilidade de acertos na detecção. Associado à detecção de curtos, implementou-se um gerador de pulsos sincronizados ao incio do curto-circuito. Os pulsos resultantes foram utilizados para sincronização do disparo da câmera **CMOS** com o fenômeno dos curto-circuitos. Obtiveram-se resultados razoáveis, com redução expressiva da saturação proveniente do arco elétrico. Entretanto, observaram-se momentos de detecção de falso curto e não detecção de alguns curtos verdadeiros. Essas falhas, entretanto, não invalidam o sistema, uma vez que a observação de sua ocorrência é de baixa freqüência se comparada aos acertos.

Para aprofundar o conhecimento sobre visão computacional, estudaram-se diversos temas relacionados, tais como: (a) como se formam imagens digitais; (b) tipos de sensores de imagem envolvidos; (c) responsividade espectral; (d) conceito de *Megapixel*; (e) banda de transmissão; (f) tipos de transmissão de dados; (g) necessidade do uso de um *frame grabber*. Este estudo teve por objetivo embasar a escolha da infraestrutura necessária ao uso do sistema industrial de captura de imagens. Este apresentava grande flexibilidade devido a suas características específicas, dentre as quais destacam-se duas: (a) o uso seletivo do sensor **CMOS**, que permite a seleção de uma região de interesse (**ROI**); e (b) a responsividade espectral compatível com aplicações na região do espectro visível e do infravermelho próximo. Considerando o fato de a poça de fusão estar a uma temperatura elevada, da ordem de 1500 ^{o}C, é de se esperar que a emissão de radiação ocorra com maior intensidade na região do infravermelho próximo. Isso motivou a busca de filtros ópticos para realçar a região da poça e eliminar ruídos indesejados. Verificou-se que, dentre os filtros disponíveis, o que apresentou melhores resultados foi o filtro passa altas para comprimento de onda (*long wave pass filter*) com corte para comprimentos de onda inferiores a 800 nm.

Outro problema enfrentado foi a especificação de uma objetiva adequada aos requisitos do sistema. Considerando que este sistema, quando totalmente desenvolvido, deve permitir sua fixação no sistema de movimentação da tocha, é de se esperar que a distância focal do conjunto não seja elevada. Além disso, é necessário que a imagem da poça seja adquirida com boa resolução para permitir sua medição com boa precisão. Tendo em vista esses fatores, buscou-se dentre as opções disponíveis a objetiva que satisfizesse os requisitos apontados. O conjunto que melhor se adaptou foi a objetiva de 50 mm associada a um espaçador de 15 mm, resultando em uma distância focal de aproximadamente de 120 mm com resolução aproximada de 300 x 300 *pixels* (dimensão aproximada da poça de fusão resultante dos parâmetros de soldagem definidos para o cordão padrão). Durante a execução dos experimentos, o uso do conjunto objetiva de 50 mm e filtro óptico infravermelho resultou em problemas, uma vez que os ajustes de foco e de abertura do diafragma são realizados com sem a luminosidade da poça fundida. Isso implica na necessidade de se utilizar iluminação auxiliar, com emissão no espectro do infravermelho próximo, para permitir a sensibilização do sensor **CMOS** da câmera. Neste trabalho, utilizou-se uma lâmpada halógena de 500 W a qual satisfez as necessidades do sistema. Deve-se enfatizar, entretanto, que a escolha desse sistema de iluminação não envolveu nenhum cálculo específico.

Para desenvolver o software de captura de imagens da poça fundida, necessitou-se analisar o funcionamento do *frame grabber*, dos *drivers* e dos *softwares* presentes na **API SAPERA LT**. Devido à complexidade do uso do sistema industrial de captura de imagens, foi necessário adquirir experiência com todas as atividades envolvidas com o uso do sistema: a instalação do *frame grabber*, a instalação da **API SAPERA LT** e a integração e o funcionamento conjuntos dos *softwares CamExpert* e *PFRemote*. Desenvolveu-se, a partir da experiência e do conhecimento adquirido, um *software* de captura e de sincronização de imagens da poça fundida. O *software* desenvolvido apresentou desempenho adequado, quando os quadros capturados são armazenados diretamente em memória **RAM** e, após a finalização do processo de captura, armazenados em disco. O problema desse algoritmo é a limitação do experimento à quantidade de memória volátil disponível no sistema computacional. Uma possível solução para este problema seria, em um futuro *software*, a implementação de 3 processos baseados em *threads* simultâneas dependentes: a captura, o processamento e o armazenamento em disco rígido.

Para desenvolver o software de análise das imagens capturadas da poça fundida, realizaram-se inicialmente testes com a **API SAPERA LT**. Este foram infrutíferos,

pois a **API** não fornecia ou não possuía funções básicas para o processamento eficiente de imagens. Como alternativa, optou-se pelo uso da **API OpenCV** (distribuída gratuitamente pela Intel). Os resultados obtidos com o uso da **API OpenCV** evidenciaram que é possível sua integração ao software de captura de imagens sem perda de eficiência. Sua inclusão no *software* desenvolvido neste trabalho permitiu o uso de uma série de funções nativas de processamento de imagens, tais como a aplicação de filtros digitais.

Para desenvolver e validar os algoritmos de processamento de imagens da poça, uma seqüência de passos foi então proposta. No primeiro passo deve-se capturar imagens em da poça de um processo de soldagem com parâmetros previamente definidos. As imagens coletadas devem ser cuidadosamente selecionadas, buscando exemplares com diversas configurações de distribuição de intensidades de brilho. Estas devem ser processadas por algoritmo de detecção de bordas previamente desenvolvidos em ambiente *MATLAB*, de modo a facilitar e a acelerar o desenvolvimento do *software*. Caso o algoritmo obtenha sucesso com as imagens selecionadas, o segundo passo é portar o algoritmo desenvolvido em *MATLAB* para C/C++ utilizando as funções da **API OpenCV** e aplicar o algoritmo de medição para todas as imagens capturadas. A analise realizada deve ser exportada para um arquivo ao final do programa. Esse passo visa a validar a eficiência computacional e a capacidade de generalização do algoritmo proposto com todas as imagens de um experimento. Caso o algoritmo obtenha sucesso com grande parte das imagens selecionadas , o terceiro é passo integrar o algoritmo em C/C++ ao software de captura e analise de imagens. O uso desses passos mostrou-se cansativo, porém seguro, pois aumenta a generalização do processo como base arquivos de experimentos anteriores.

Inicialmente, a análise de imagens foi realizada conforme algoritmos propostos em trabalho anterior (Koike 1999). Estes, entretanto, apresentavam desempenho inadequado para utilização em altas taxas de aquisição de imagens, devido ao elevado tempo de processamento necessário por imagem. Analisando esses algoritmos e os tipos de imagens características do processo em questão, propuseram-se e implementaram-se alterações nos algoritmos de medição, de modo a aumentar sua eficiência computacional. As alterações implementadas mostraram-se eficientes e eficazes, tendo em vista o fato de as imagens adquiridas com o uso do sistema de sincronização apresentarem-se de forma mais consistente, do ponto de vista da distribuição de intensidade de brilho. Entretanto, observou-se que o algoritmo de sincronização necessita um ajuste fino, tendo em vista o fato de ainda serem adquiridas imagens com saturação na extremidade do

arame, provavelmente devido à captura ter ocorrido antes da extinção completa do arco. O algoritmo de medição utiliza as transições bruscas de preto e branco para determinar a posição das bordas da poça e do ponto de toque do arame. Deve-se lembrar que, de acordo com algoritmo apresentado na seção 3.5.7, a largura da poça é medida na região horizontal em torno da posição de toque do arame. Isso implica que variações na posição do ponto de toque, decorrentes de presença de saturação do arco, aumentam a incerteza na medida da largura da poça resultante, introduzindo, conseqüentemente, ruido no sinal produzido. O problema provavelmente pode ser resolvido por meio da redução do tempo de exposição do sensor na captura das imagens, ou por meio da introdução de um atraso na geração do pulso após a detecção do curto-circuito. Outra possível solução seria a utilização da resistência dinâmica como sinal para detecção de curto-circuitos, ao invés da tensão de soldagem. Esta última não foi testada neste trabalho.

5 CONCLUSÕES

O objetivo desse trabalho foi produzir um sistema de visão computacional integrado a um sistema de medição de parâmetros de soldagem a fim de capturar imagens da poça de soldagem com pouco ou nenhum brilho, para então analisar alguns parâmetros dessa poça como: o tamanho da poça, o alinhamento do arame, posição media da poça. Todo esse processamento deve ocorrer aproximadamente em tempo de soldagem.

Os resultados dos experimentos permitiram concluir:

- É possível desenvolver um sistema de visão computacional, não intrusivo, para monitorar um processo de soldagem **GMAW** por curto-circuito;
- O uso da técnica de janelamento, que analisa a tensão elétrica produzida em um processo de soldagem **GMAW** por curto-circuito, permite sincronizar a captura de imagens aos períodos de menor intensidade luminosa do arco de soldagem;
- As câmeras **CMOS** permitem altas taxas de aquisição de imagens, além de possibilitarem a definição de regiões de interesse.
- Filtros ópticos passa alta para comprimento de onda, com freqüência de corte na região do infravermelho próximo, reduzem o brilho do arco elétrico e evidenciam a poça de fusão em um processo de soldagem;
- A **API SAPERA LT** é eficiente na captura e ineficiente no processamento de imagens, portanto deve-se utilizar a **API Open CV** como alternativa para processar imagens.
- Algoritmos de medição da poça fundida apresentados neste trabalho são eficientes em altas taxas de captura, porém são imprecisos para medir a poça em períodos de formação ou quando as imagens capturadas estão sem foco ou com saturação excessiva.

Caso o sistema de visão apresentado nesse trabalho seja melhorado e laçado comercialmente, este abrirá um leque de projetos a serem desenvolvidos, entre eles, a possibilidade de monitorar e registar processos de soldagem industriais robotizados.

5.1 Trabalhos futuros

Um dos projetos futuros, seria a implementação de um controle em malha fechada, que utiliza como realimentação os dados coltados pelo sistema de visão computacional aliados a sinais tradicionais como a tensão elétrica do processo de soldagem, para controlar um braço robotizado e a fonte de soldagem a fim de compensar defeitos de soldagem detectados em tempo de execução.

Outros projetos a serem realizados envolvem a melhoria deste trabalho como: (a) implementação de um sistema que controle automaticamente o diafragma e foco da câmera baseado em redes neurais; (b) portar o *software* de captura e análise para um ambiente **FPGA**; (c) desenvolver um *software* divido em *threads* para que seja possível, simultâneamente, capturar, analisar e armazenar em disco um experimento.

REFERÊNCIAS BIBLIOGRÁFICAS

ADOLFSSON, S. et al. On-line quality monitoring in short-circuit gas metal arc welding. *Welding Journal*, p. no2 59s–73s F, 1999.

ADVANTECH. *QA Test Report*. : ADVANTECH CO LTD, 2003.

AGAM, G. Introduction to programming with opencv. *Department of Computer Science, Illinois Institute of Technology*, Janeiro 2006.

ANDERSEN, K.; JOEL, R.; COOK, G. Weldsmart a vision-based expert system for quality control. *Mid-South Engineering Inc*, 1992.

BALFOUR, C.; SMITH, J.; AL-SHAMMA, A. A novel edge feature correlation algorithm for real-time computer vision-based molten weld pool measurements. *Welding Journal*, p. 1–s 8–s, 2006.

CAMERA LINK. Specifications of the camera link interface standard for digital cameras and frame grabbers. Outubro 2000.

CARLSON, N. et al. Sensing the gas metal arc welding process. *Idaho National Enginieering Laboratory*, 1993.

CARVALHO, G. C. *An adaptive control system for off-line programming in robotic gas metal arc welding*. Tese (Doutorado) — University of Cranfield, CITECH, Inglaterra, 1997.

DALSA. *Sapera pixel processor module programmer's manual*. 5. ed. 7075 Place Robert-Joncas, Suite 142, St-Laurent, Quebec, H4M 2Z2, Canada: Coreco Inc, 2003.

DALSA. *Sapera LT 6.0 Acquisition Parameters Reference Manual*. 2. ed. 7075 Place Robert-Joncas, Suite 142, St-Laurent, Quebec, H4M 2Z2, Canada: Coreco Inc, 2006.

DALSA. *Sapera LT ActiveX Manual*. 6.00. ed. 7075 Place Robert-Joncas, Suite 142, St-Laurent, Quebec, H4M 2Z2, Canada: Coreco Inc, 2006.

DALSA. *Sapera LT Basic Modules Reference Manual*. 6.00. ed. 7075 Place Robert-Joncas, Suite 142, St-Laurent, Quebec, H4M 2Z2, Canada: Coreco Inc, 2006.

DALSA. *Sapera LT Users Manual*. 6.00. ed. 7075 Place Robert-Joncas, Suite 142, St-Laurent, Quebec, H4M 2Z2, Canada: Coreco Inc, 2006.

DALSA-CORECO. *X64-CL iPro series users manual*. 1.11. ed. 7075 Place Robert-Joncas, Suite 142, St-Laurent, Quebec, H4M 2Z2, Canada: [s.n.], 2006.

DIGI INTERNATIONAL. *PCI Technology Overview*. : [s.n.], 2003.

FRANCO, L. D. N. *Sistema de Apoio e Monitoraçao de Veículos Baseado em Posicionamento Global por Satélite e em Mensagens SMS*. : IESB, 2005.

GRANJA, T. V. *Desenvolvimento de estratégias de condicionamento de sinais provenientes de processo de soldagem*. Brasília, 2001.

KOIKE, C. C. Monitoração da poça de fusão no processo gmaw. *Congresso Brasileiro de Engenharia Mecânica (CDROM)*, n. XV, p. 22–26, Novembro 1999.

LITWILLER, D. Ccd vs cmos: facts an fiction. *Photonics Spectra*, 2001.

MARQUES, P.; MODENESI, P.; BRACARENSE, Q. *Soldagem fundamentos e tecnologia*. Belo Horizonte: Editora UFMG, 2005.

NORRISH, J. *Advanced welding processes*. Institute of Physics, Bristol , Inglaterra: IOP, 1992.

PISAREVSKY, V. Introduction to opencv. Junho 2007.

RIBEIRO, A.; FIGUEIREDO, M. *Tutorial de OpenCV para Tótós*. Instituto Superior Técnico, Universidade Técnica de Lisboa, Portugal, 2005.

TOGAWA, E. T. *Sistema de Monitoração de Sinais para Processos de Soldagem Robotizada*. Universidade de Brasília, 2003.

ZHENGUO, S. et al. A novel visual image sensor for co_2 short circuiting arc welding and its application in weld reinforcement detection. *Measurement Science and Technology*, v. 17, p. 3212 – 3220, 2006.

ANEXOS

A Manual de utilização do sistema de visão

Para utilizar o sistema computacional de visão para monitorar um processo de soldagem **GMAW** por curto-circuito é necessário levantar se todos esses materiais abaixo estão disponíveis:

I - fonte de soldagem

1.Fonte de soldagem e conexões elétricas

2.Mistura de gás argônio 75 e carbono 25

3.Tubulações conectores e manômetro

4.Verficar se existe água desmineralizada dentro da máquina de soldagem

5.Suporte das tocha de soldagem

6.Alimentador de arame de soldagem e Bobina

7.Tocha de soldagem com bocal limpo e grafitado

8.Cabo de obra

II - Mesa Linear

1.Fonte de 24 V DC para alimentação do Motor e conexões elétricas

2.Caixa preta de controle da mesa

3.Cabo serial com TX/RX e GND padrão e cabo de controle da mesa Linear

4.Motor de passo

5.Eixo cardan

6.Mesa linear

7.Protetor do fuso da mesa

8.Placa de baquelite e parafusos

9.Software de controle da mesa em Labview

III - Sistema de medição da tesão arco do soldagem e Sincronização

1.2 Fontes de alimentação simétricas isoladas +- 12V e conexões elétricas

2.Placa para redução da tensão do arco elétrico e cabos de conexão

3.Placa de isolamento da Tensão Elétrica

4.Placa de detecção do curto-circuito e cabos de interconexão

5.Cabo de ligação entre a placa de detecção do curto-circuito e o *framegrabber*

6.Programa de compilação para microcontroladores PICC (pic c compiler)

7.Programa de gravação em microcontroladores WinPic 800

8.Software de detecção de curto-circuito em microcontrolador em C

IV - Sistema de Visão

1.Câmera Dalsa DS-21-001M150 e e conexões elétricas

2.Cabo CameraLink de 5m para a câmera

3.Objetiva de 25mm e anilhas espaçadoras para as objetivas

4.Caixa protetora da Câmera

5.Placa de vidro para proteger a lente de respingos

6.Filtro óptico infravermelho passa alta ¿= 800 nm

7.Suporte para filtro óptico

8.Parafusos e anilhas para colocação da câmera e filtro na caixa protetora

9.Suporte para fixar a caixa protetora na tocha de soldagem e parafusos

10.Holofote de 500W

11.Computador industrial com Sistema operacional windowsXP SP2

12.Software Sapera LT 6.X instalado

13.Driver do Framegraber instalado

14.Software Camexpert e PFRemote instalado

15.API OpenCV instalada

16.Visual Studio 2005 instalado

17.Matlab instalado

18.Software de captura e analise de imagens de imagens em C/C++

V - Sistema de aquisição de dados

1.Computador industrial com Sistema operacional windowsXP SP2

2.Placa de aquisição de dados PCI-703S-16A

3.Driver da placa de aquisoção mais a API de desenvolvimento para LabView

4.Labview 8.0 instalado

5.Placa interconexão entre placa de aquisição e dispositivos diversos

6.Cabo de ligação entre a placa de aquisição e a placa interconexão

7.Software de aquisição de dados em Labview

Para montar o sistema de monitoração deve ser montado na seqüência I, II, III, IV, V.

Montagem do Sistema I - fonte de soldagem

1.Ligar a fonte de soldagem na rede elétrica (cuidado com a Fase)

2.Conectar a garrafa de gás no manômetro

3.Conectar a tubulação no manômetro e na máquina de soldagem

4.Conectar a tocha de soldagem no alimentador não esquecer água, gás e arame

5.Fixar a tocha de soldagem no suporte

Montagem do Sistema II - Mesa Linear

1. Acoplar e fixar: a mesa linear, eixo cardan e motor de passo
2. A montagem da mesa deve ser em local nivelado para não afetar a soldagem
3. Fixar a placa de baquelite na mesa
4. Fixar o protetor de fuso
5. Ligar a fonte de 24V na caixa preta de controle da mesa
6. Conectar cabo serial no computador e cabo de controle da mesa Linear na caixa preta de controle da mesa
7. Testar Software de controle da mesa em Labview

Montagem do Sistema III - Sistema de medição da tesão arco do soldagem e Sincronização

1. Acoplar os cabos do redutor de tensão positivo no alimentador (vermelho) e negativo (preto) na obra
2. Ligar esses cabos na placa de redução da tensão do arco elétrico
3. Ligar a saída da placa de redução da tensão na placa de isolamento da Tensão Elétrica no texto "IN"
4. Ligar a saída da placa de isolamento da Tensão Elétrica "OUT" na placa de detecção de curto-circuito (encaixe duplo próximo ao conector do cabo de rede)
5. Ligar a saida da placa de detecção de curto-circuito (cabo de rede) ao cabo de ligação entre a placa de detecção do curto-circuito e o *framegrabber* (parece com um conector de cabo parelelo e um conector de placa de rede) CUIDADO COM A POLARIDADE + e - caso tenha dúvidas se o cabo está correto (use o o multímetro e o manual da placa ja que o cabo não possui nehuma indicação)
6. Alimentar a placa de detecção de curto-circuito com uma fonte de 9V e alimentar a placa de isolamento da tensão elétrica com tensões +- 12 provenitente de fontes simétricas e isoladas entre si!
7. Para testar de forma offline a detecção do curto-circuito insira um sinal senoidal na porta de rede e compare com um osciloscópio com a porta de saida

8. Para testar de forma online a detecção do curto-circuito a câmera e o programa de captura devem ser utilizados

Montagem do Sistema IV - Sistema de Visão

1. Montar a câmera anilhas espaçadoras e objetiva com cuidado
2. Fixar a câmera na caixa protetora
3. Fixar o filtro óptico e utilizar as anilhas para alinhar a objetiva e o filtro na caixa protetora
4. Fixar o vidro de proteção por meio de uma fita crepe
5. Fixar o suporte na tocha de soldagem
6. Fixar a caixa protetora no suporte
7. Conectar cabo câmera Link e fonte da câmera CUIDADO pode dar Choque
8. Conectar o cabo no *framegrabber*
9. Ligue o holofote de 500W e aponte, incialmente, diretamente para a câmera
10. Abrir os programas CamExpert e PFRemote
11. No programa *CamExpert* selecione em que porta do *framegrabber* a câmera está instalada (em Device) e o modelo da câmera (em camera) e clique em *grab* (a câmera só funciona em uma porta do *framegrabber*)
12. No programa *PFRemote* encontre onde a câmera está conectada e na aba *Exposure* time no campo *Exposure* time altere a barra para um um valor próximo de 10 fps
13. No programa *CamExpert* se algo diferente do preto aparecer quer dizer que a câmera está ok caso não de certo tente alterar o diafragma da objetiva (pode estar totalmente fechado) ou retire o filtro óptico e mexa um pouco com a câmera até obter uma certa experiência
14. Para ajustar o foco e abertura do diafragma deve-se estar ciente que as anilhas reduzem o foco da objetiva para 12 cm, por isso, aponte a câmera para um objeto claro ou metálico próximo a objetiva e mexa no holofote até obter algo parecido com objeto é possível que o objeto não esteja em direção da câmera.

15. Altere primeiramente o foco e ajuste o diafragma até o que o objeto esteja bem evidente e a imagem esteja ligeiramente claro

16. O sistema está pronto para ser testado com o sistema de captura

17. Feche o programa *CamExpert* e no programa *PFRemote* reduza o tamanho da janela a ser capturada na aba window segundo o texto 3.5.1 Subutilização do sistema de captura

18. Abra o programa Visual Studio 2005 abra o projeto de captura de imagens (SaperaOpenCV_2_0) disponível no DVD desse trabalho

19. Compile o programa e escolha a opções como tempo (não exagere mais de 30 s) e inicie a captura ao mesmo tempo que da realização da soldagem

20. Observe os resultados na pasta de origem do projeto

B Manual de montagem de um computador industrial

Um computador industrial deve ser robusto e de fácil manutenção e escalabilidade, por isso difere de um computador doméstico. A primeira e mais evidente é o gabinete.

O gabinete computador utilizado nesse trabalho Figura B.1 é produzido pela empresa ***Advantech*** modelo **ICP-622** possui um série de melhorias em relação de uma gabinete comum. A primeira é que as chapas de construção utilizadas nesse gabinete são muito mais espessas podendo a medir até 3 mm Figura B.1 (A).

A utilização de contatos para aterramento cromados em toda parte do gabinete Figura B.1 (B). Isso melhora a imunidade a ruídos causados por picos da rede ou motores, fontes de soldagem próximas ao computador.

Para evitar a entrada de partículas nocivas à máquina, existe um conjunto de quatro ventiladores de alta velocidade associados a filtros que criam um sistema ventilação forçada que propicia a formação de pressão positiva no interior do gabinete definindo um único local para entrada de ar que possui um filtro de ar Figura B.1 (C) e (B) evitando a formação de poeira interior do gabinete e aumentando o tempo entre manutenções preventivas.

E não mais importante é a utilização de fontes redundantes de alta potência 460W Figura B.1 (D) que caso existam fontes de alimentação em ramais diferentes e caso uma venha a cair ou queimar a outra assume e um alarme é acionado podendo então um operador substituí-la pela outra sem desligar o sistema por meio de uma substituição a quente.

Um computador doméstico possui uma placa mãe que nela estão encaixados processador memória, placa de vídeo cabos do disco rígido se perceber na Figura B.2 (uma ampliação Figura B.1) não existe nada disso descrito anteriormente somente uma série de barramentos. Na verdade realmente isso não é uma placa mãe isso se chama *back-plane* esse dispositivo é uma placa passiva ou seja que possui somente entrada para alimentação e barramentos.

Note na Figura B.2 que existem 4 conjuntos (1 2 3 4) independentes, cada qual com

Figura B.1: Detalhes de um gabinete industrial

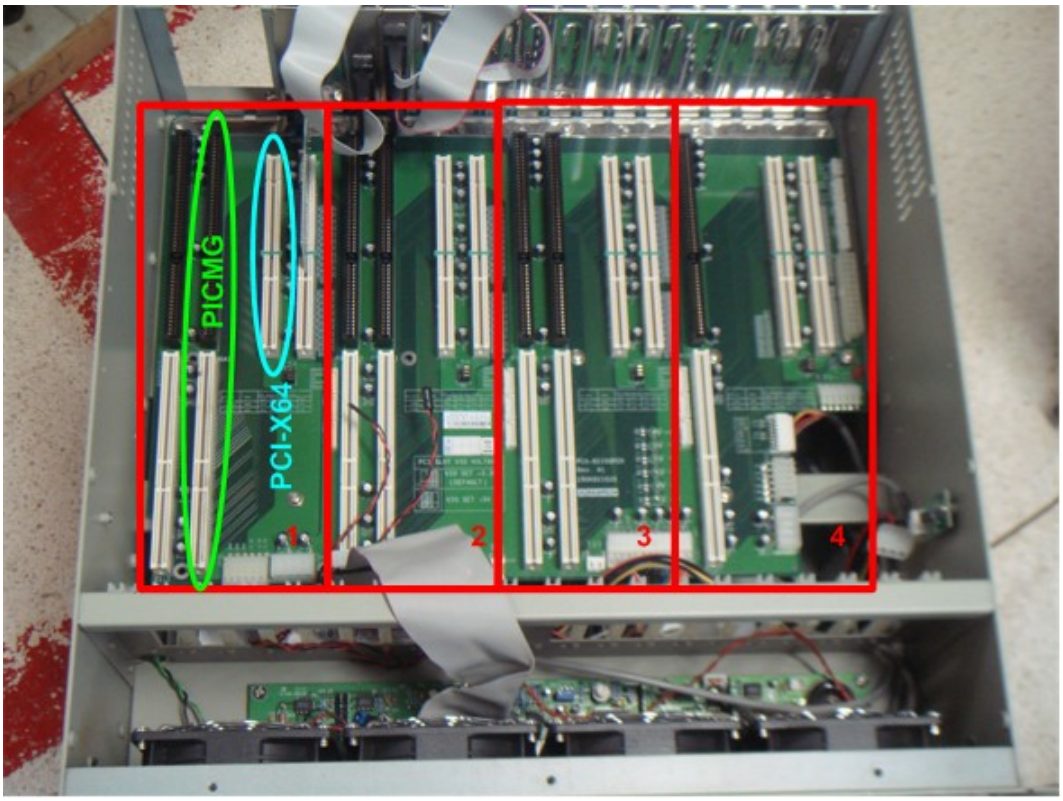

Figura B.2: *Backplane* em detalhe com quatros placas independentes

dois barramentos **PICMG** (*industrial computer manufacturers group*) e o **PCI-X64**. O *backplane* utilizado nesse trabalho é produzido pela ***Advantech*** modelo PCA-6115QP2X.

O barramento **PICMG** assim como o **CAMERA LINK** é um consórcio de empresas com o objetivo de produzir equipamentos de alto desempenho para áreas de telecomunicações, médicas, militares e industriais (www.picmg.org). Esse barramento pode possuir, dependendo do tipo de placa, a compatibilidade com os obsoletos barramentos ISA 8 e 16 bits e os mais recentes o **PCI** e **PCI-X64**

No barramento **PICMG** é local onde deve ser instalado a placa mãe que deve ser específica para esse tipo de barramento, por isso, a idéia de criar vários sistemas de monitoração de soldagem, já que, o *backplane* possui capacidade de manter até 4 sistemas de monitoração independentes.

As placas mães próprias ao barramento **PICMG** são placas altamente integradas pois nelas são colocadas todos os sistemas de um computador doméstico em uma única placa de tamanho reduzido e desempenho otimizados tanto para durabilidade quanto processamento.

Os componentes mais comuns desse tipo de placa são *sockets* para processador e memória, controladoras **FDDI**, **IDE**, **USB**, **RS-232** e **SCSI**, além disso, possui controladores de video, rede. A placa **PICMG** utilizada nesse sistema é produzida pela ***Advantech*** modelo PCA-6185 Figura B.3 que pode trabalhar com processadores da linha *Pentium* 4 anteriores ao **HT** com memória **DDR** registrada podendo chegar até 4GB, além disso integra um placa de video de 8 MB, 2 portas de rede *GigaBit Ethernet*, portas **USB** 1.1, RS-232, Paralela, os tradicionais barramentos **IDE** e **FDDI** e o veloz barramento *ultra wide* ***SCSI*** de 320 MB/s.

Nesse trabalho a configuração utilizada nessa placa foram: um processador *Pentium* 4 2.8 GHz, um pente de memória **RAM** de 1GB.

Uma das dificuldade em se utilizar um computador industrial é a dificuldade em se encontrar equipamento e suporte especializado no Brasil, para montagem dessa placa **PICMG**, em particular, que utiliza memória **DDR** registrada foi necessário importar esse componente diretamente do fabricante por meio de um representante nacional com um tempo médio de espera de aproximadamente 2 meses, outro problema foi encontrar o processador para esse tipo de equipamento que já é obsoleto, apesar do desempenho, pois era um modelo de *Pentium* 4 com o tipo de *socket* de processador mais antigo.

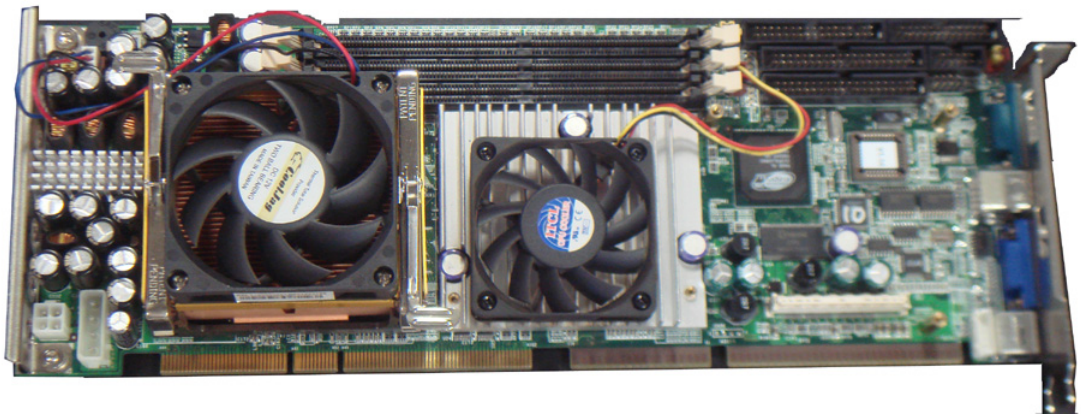

Figura B.3: Placa **PICMG** note o nível de integração

O *frame grabber* utilizado nesse trabalho produzido pelo fabricante **DALSA** modelo **X64-CL iPro**^{TM} que utiliza a tecnologia de barramento **PCI-X64** com capacidade máxima de 533 MB/s para dois canais.

O computador a ser montado para o sistema de visão utilizado nesse trabalho que é composto de um gabinete industrial, um *backplane*, uma placa **PICMG**, uma placa **PCI-X64**, um disco rígido **IDE** de 250 GB, cabos diversos e uma pulseira antiestática afim de evitar a queima prematura de componentes **CMOS** sensíveis a eletricidade estática.

Os passos evidendiados na Figura B.4 para montar este computador foram instalar primeiramente o disco rígido (A), e o *backplane* no gabinete, instalar a placa do *frame grabber* (B), e depois instalar delicadamente a placa **PICMG** com os componentes discretos já nela montados (processador, memória, *cooler*) (C), ligar os cabos **IDE** a placa **PICMG**, ligar os cabos restantes como portas seriais e paralelas adicionais e e portas de *trigger* do *frame grabber* (D).

Os empecilhos encontrados durante a montagem foram definir o posicionamento adequando da placa **PICMG** no gabinete em relação ao disco rígido pois o cabo **IDE** fornecido pelo fabricante é muito curto Figura B.4 (D). Para instalação da placa **PICMG** foi necessário serrar parte do gabinete afim de obter o espaço adequado para a placa que ocupa 2 *slots* Figura B.1 (A).

Após a montagem do computador deve-se instalar um sistema operacional segundo o fabricante da **API** do *frame grabber* deve-se utilizar um sistema operacional baseado em sistemas *Microsoft Windows* com sistemas operacionais *Windows 2000* ou superior.

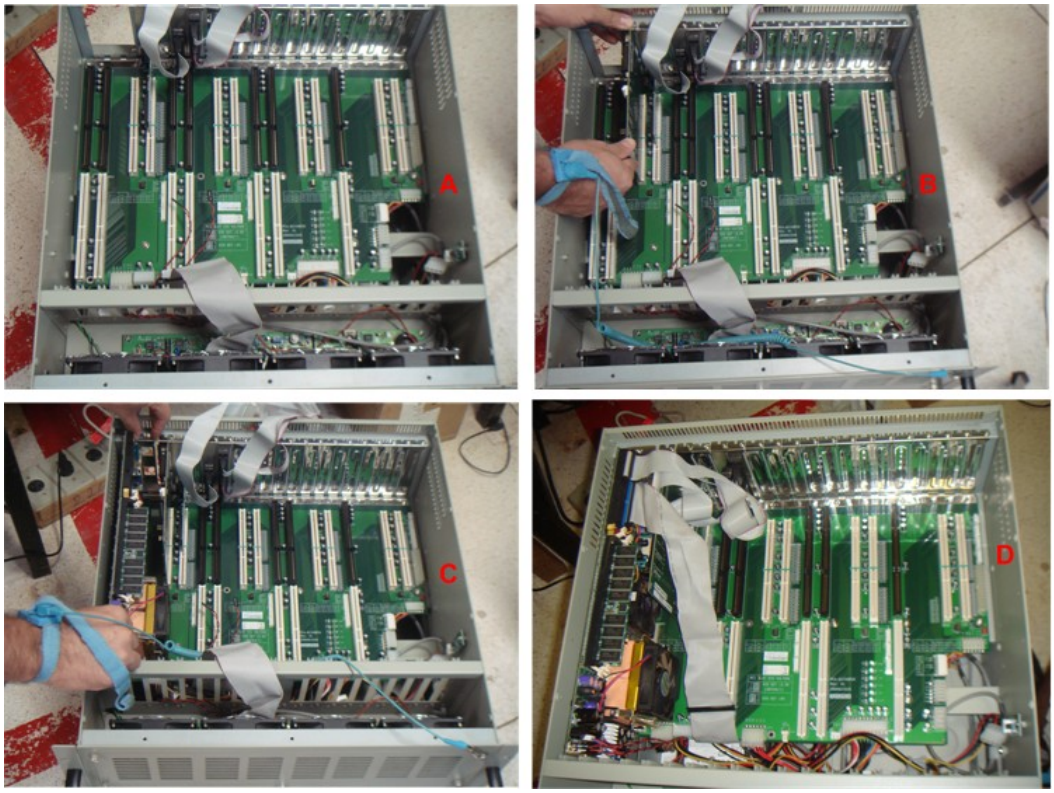

Figura B.4: Passos para instalar um computador industrial

Devido a disponibilidade de mercado só foi possível adquirir um sistema operacional licenciado *Microsoft Windows XP SP1 - Português*.

Durante a instalação do sistema operacional na placa **PICMG** com as configurações default foi detectado a impossibilidade de instalação direta do sistema operacional de origem desconhecido (suspeita-se de alguma incompatibilidade com a controladora **SCSI**).

Consultando o suporte técnico do representante brasileiro não foram obtidas soluções adequadas até que o Sr. Luis Toledo, com larga experiência em sistemas industriais, sugeriu que primeiro fosse instalado um sistema operacional *windows 98 second edition* ou *windows 2000* para depois atualizar, via *windows*, para o sistema operacional o *windows XP*, solucionando assim o problema.

Problemas de obsolescência ocorrem em qualquer tipo de sistema computacional, porém para sistemas industriais desde que o equipamento seja eficiente, robusto, e que possua uma manutenção viável o equipamento continuará sendo útil por até décadas. O equipamento utilizado nesse trabalho hoje já é considerado obsoleto (esta tese foi escrita e 2007), pois esses equipamentos foram lançados em 2002/2003 e estão fora de

linha segundo o fabricante, mas continua a oferecer algumas peças de reparo, porém no Brasil não fazem reparo direto na placa **PICMG**.

A grande vantagem de se possuir um equipamento industrial apesar do fabricante,no Brasil, não oferecer mais suporte a é a existência um grande **MTBF** (mean time between faliure) associado a essa placa que é de 54.608 horas (ADVANTECH 2003) isso significa 6 anos de uso sem interrupções. Na ocorrência de falha o fabricante provavelmente na época irá oferecer uma placa que possa substituir esses sistemas sem grandes perdas de produção.

C Instalação da API SaperaLT no Visual Studio 2005

Criar um Project

Entre em arquivo New Project na janela do wizard clique nas opções *Visual C++ - Win32 - Win32 Console Application* coloque um nome para o projeto na caixa *Name*

Em uma nova janela clique um *Next* marque as opções *Console Application* e *Empty Project* e clique em *Finish*

Criando um Arquivo Cpp

Com o projeto vazio criado na aba *Sources* e com o botão direito do mouse em *Add - new item*

Um janela será criada clique na opção *Code* e escolha um arquivo com extensão *C++ File (.CPP)* e de um nome a esse arquivo

Colocando arquivos de *Inculde*

Clique no arquivo criado com o o botão esquerdo no menu *Project* e selecione o item *Proprieties*

Uma nova janela será aberta selecione o item C/C++ e abra a arvore no símbolo + e selecione o item general e clique sobre o "..." na caixa *Additional Include Directory* uma nova janela será aberta basta clicar sobre o ícone de uma pasta para criar um novo diretório para que seja compilado junto ao projeto clique novamente nos "..." e selecione o diretório

```
{C:\DALSA Coreco\Sapera\Lib}
```

faça o mesmo procedimento e coloque

{C:\DALSA Coreco\Sapera\Include}

clique em ok duas vezes

Adicionado Arquivos necessários ao projeto

Na Aba *Header Files* e clique com o botão Direito selecione *Add - Existing Item* uma nova janela será aberta então vá para o diretório

C:\DALSA Coreco\Sapera\Include}

e selecione o arquivo corapi.h

Na aba *Resource File* e clique com o Botão Direito Add - Existing Item uma nova janela será aberta então vá para o diretório

{C:\DALSA Coreco\Sapera\Lib}

mude o tipo de arquivo para *Type File* para *All Files* e selecione o arquivo corapi.lib Caso apareça uma pergunta basta marcar *No*

Agora clique no arquivo .cpp e inclua ao arquivo fonte todo o conteúdo abaixo e faca um teste ... com o arquivo de configuração de câmera já definido

Programa exemplo do manual **API SAPERA** *BASIC*

```
#include <stdio.h>
#include <corapi.h>

// Transfer callback function: called each time a complete frame is transferred
//
CORSTATUS CCONV XferCallback (void *context, UINT32 eventType, UINT32 eventCount)
{
// Display the last transferred frame
CorViewShow(*(CORVIEW*) context);
return CORSTATUS_OK;
}

//
// Example program
//
main()
```

```
{
CORSTATUS status; // Error code
CORSERVER hSystem; // System server handle
CORSERVER hBoard; // Board server handle
CORCAM hCam; // CAM handle
CORVIC hVic; // VIC handle
CORACQ hAcq; // Acquisition handle
CORBUFFER hBuffer; // Buffer handle
CORXFER hXfer; // Transfer handle
CORVIEW hView; // View handle
CORDISPLAY hDisplay; // Display handle
UINT32 width, height, format;
// Initialize Sapera API
status = CorManOpen();
// Get server handles (system and board)
hSystem = CorManGetLocalServer();
status = CorManGetServerByName("X64-CL_1", &hBoard);
// Get acquisition handle
status = CorAcqGetHandle(hBoard, 0, &hAcq); // 0 = First instance
// Create CAM/VIC handles
status = CorCamNew(hSystem, &hCam); // Camera
status = CorVicNew(hSystem, &hVic); // Video-Input-Conditionning
// Load CAM/VIC parameters from file into system memory
// The acquisition hardware is not initialized at this point
status = CorCamLoad(hCam, "rs170.cca");
status = CorVicLoad( hVic, "rs170.cvi");
// Download the CAM/VIC parameters to the acquisition module
// The acquisition hardware is now initialized
status = CorAcqSetPrms(hAcq, hVic, hCam, FALSE);
// Create a buffer compatible to acquisition
status = CorAcqGetPrm(hAcq, CORACQ_PRM_SCALE_HORZ, &width);
status = CorAcqGetPrm(hAcq, CORACQ_PRM_SCALE_VERT, &height);
status = CorAcqGetPrm(hAcq, CORACQ_PRM_OUTPUT_FORMAT, &format);
status = CorBufferNew(hSystem, width, height, format,
CORBUFFER_VAL_TYPE_SCATTER_GATHER, &hBuffer);
// Create a transfer handle to link acquisition to buffer
status = CorXferNew(hBoard, hAcq, hBuffer, NULL, &hXfer);
// Register a callback function on "End-Of-Frame" events
status = CorXferRegisterCallback(hXfer, CORXFER_VAL_EVENT_TYPE_END_OF_FRAME,
XferCallback, (void *)&hView);
// Activate the connection between acquisition and buffer
status = CorXferConnect(hXfer);
// Get display handle
status = CorDisplayGetHandle(hSystem, 0, &hDisplay);
// Create a view handle and assign it to a HWND
status = CorViewNew(hSystem, hDisplay, hBuffer, CORVIEW_VAL_MODE_AUTO_DETECT,
&hView);
status = CorViewSetPrm(hView, CORVIEW_PRM_HWND, -1); // -1: create new window
// Start a continuous transfer (live grab)
status = CorXferStart(hXfer, CORXFER_CONTINUOUS);
printf("Press any key to stop grab\n");
getchar(); // wait until a key has been hit
// Stop the transfer and wait (timeout = 5 sec)
status = CorXferStop(hXfer);
status = CorXferWait(hXfer, 5000);
// Break the connection between acquisition and buffer
status = CorXferDisconnect(hXfer);
printf("Press any key to terminate\n");
getchar();
// Release handles when finished (in the reverse order)
status = CorViewFree(hView);
status = CorDisplayRelease(hDisplay);
status = CorXferFree(hXfer);
status = CorBufferFree(hBuffer);
status = CorVicFree(hVic);
status = CorCamFree(hCam);
status = CorAcqRelease(hAcq);
// Close Sapera API
status = CorManClose();
return 0;
}
```

D Código fonte de captura de imagens 1.0

```
#include <stdio.h>

#include <time.h>

#include <corapi.h>

#define TEMPO_DE_AQUISICAO 10000 //Tempo em ms
#define NUMERO_DE_QUADROS 6000   //Número máximo de quadros que poderiam ser amostrados

CORSTATUS status;     // Error code
CORBUFFER hBuffer;    // Buffer handle
CORBUFFER hFilho[NUMERO_DE_QUADROS];    // Buffer Filho

int countQuadros =0; //Contador de quadros

LARGE_INTEGER frequencyPerformancy;  //Recebe qunatos ticks o sistema tem por segundo
unsigned long double iniciaCaptura=0,fimCaptura=0;
unsigned long double tempos[NUMERO_DE_QUADROS];


unsigned long double GetTimeMs()
{
   LARGE_INTEGER m_large_int_tmp_init;
   QueryPerformanceCounter( &m_large_int_tmp_init );
   return ( m_large_int_tmp_init.QuadPart*1000 )/( frequencyPerformancy.QuadPart*1.00 );
   //m_lastElapsedTime = ( ( m_finalTime - m_initialTime )*1000 )/( m_frequencyPerformance*1.00 );
}


// Transfer callback function: called each time a complete frame is transfer
CORSTATUS CCONV XferCallback (void *context, UINT32 eventType, UINT32 eventCount)
{
  // Display the last transferred frame
 //   CorViewShow(*(CORVIEW*) context);
status = CorBufferCopy(hBuffer,0,0,hFilho[countQuadros]);
tempos[countQuadros] = GetTimeMs() - iniciaCaptura;
countQuadros++;

    return CORSTATUS_OK;
}

int main()
{
  CORSERVER hSystem;    // System server handle
  CORSERVER hBoard;     // Board server  handle
  CORCAM hCam;          // CAM           handle
  CORVIC hVic;          // VIC           handle
  CORACQ hAcq;          // Acquisition   handle
  CORXFER hXfer;        // Transfer      handle
  CORFILE hFile;        // File      handle

  // CORVIEW hView;        // View handle
 // CORDISPLAY hDisplay;  // Display handle
  UINT32 width, height, format;

  FILE * pFile;
  time_t seconds;

  //Captura qunatos Ticks por segundo
  QueryPerformanceFrequency( &frequencyPerformancy );

  // Initialize Sapera API
  status = CorManOpen();

  // Get server handles (system and board)
  hSystem = CorManGetLocalServer();
  status = CorManGetServerByName("X64-CL_iPro_1", &hBoard);
```

```
  // Get acquisition handle
  status = CorAcqGetHandle(hBoard, 0, &hAcq);  // 0 = First instance

  // Create CAM/VIC handles
  status = CorCamNew(hSystem, &hCam);  // Camera
  status = CorVicNew(hSystem, &hVic);  // Video-Input-Conditionning

  // Load CAM/VIC parameters from file into system memory
  // The acquisition hardware is not initialized at this point
  status = CorCamLoad(hCam, "cam_128_128_centrada.ccf");
  status = CorVicLoad( hVic, "cam_128_128_centrada.ccf");

  // Download the CAM/VIC parameters to the acquisition module
  // The acquisition hardware is now initialized
  status = CorAcqSetPrms(hAcq, hVic, hCam, FALSE);

  // Create a buffer compatible to acquisition
  status = CorAcqGetPrm(hAcq, CORACQ_PRM_SCALE_HORZ, &width);
  status = CorAcqGetPrm(hAcq, CORACQ_PRM_SCALE_VERT, &height);
  status = CorAcqGetPrm(hAcq, CORACQ_PRM_OUTPUT_FORMAT, &format);
  status = CorBufferNew(hSystem, width, height, format,
    CORBUFFER_VAL_TYPE_CONTIGUOUS, &hBuffer);

  for(int i=0;i<NUMERO_DE_QUADROS;i++)
  {
status = CorBufferNew(hSystem, width, height, format,
    CORBUFFER_VAL_TYPE_SCATTER_GATHER,&hFilho[i]);
 status = CorXferNew(hBoard, hAcq, hFilho[i], NULL, &hXfer);
  }

  // Create a transfer handle to link acquisition to buffer
  status = CorXferNew(hBoard, hAcq, hBuffer, NULL, &hXfer);

  // Register a callback function on "End-Of-Frame" events
  status = CorXferRegisterCallback(hXfer, CORXFER_VAL_EVENT_TYPE_END_OF_FRAME,
      XferCallback, (void *)&hBuffer);

  // Activate the connection between acquisition and buffer
  status = CorXferConnect(hXfer);

  // Get display handle
 // status = CorDisplayGetHandle(hSystem, 0, &hDisplay);

  // Create a view handle and assign it to a HWND
  //status = CorViewNew(hSystem, hDisplay, hBuffer, CORVIEW_VAL_MODE_AUTO_DETECT,
  //    &hView);
 // status = CorViewSetPrm(hView, CORVIEW_PRM_HWND, -1); // -1: create new window

  // Start a continuous transfer (live grab)
  status = CorXferStart(hXfer, CORXFER_CONTINUOUS);

 // printf("Press any key to stop grab\n");
  //getchar(); // wait until a key has been hit
  iniciaCaptura = GetTimeMs();
  Sleep(TEMPO_DE_AQUISICAO);
  // Stop the transfer and wait (timeout = 5 sec)
 // fimCaptura = GetTimeMs();
  //fimCaptura = GetTimeMs();

 // printf("%Le",fimCaptura);
  status = CorXferStop(hXfer);
  //status = CorXferWait(hXfer, 1000);


  //Salva Figuras / Video / Tempos
  char str [80],str2[80];// String que cria arquivo
  seconds = time (NULL);
  system("md Resultados");
  pFile = fopen (".\\Resultados\\Tempos.txt","w");
  for(int i=0;i<countQuadros;i++)
  {
    (void) sprintf(str, ".\\Resultados\\Img%i.bmp",i);

//Forma mais Atual
status = CorFileNew(hSystem,str,"wb",&hFile);
```

```
    status = CorFileSave(hFile, (CORHANDLE) hFilho[i],"-f corfile_format_bmp");

//Forma Obsoleta
//status = CorBufferSave(hFilho[i], str, CORFILE_VAL_FORMAT_BMP);

(void) sprintf(str, "Img%i.bmp",i);
//Escreve tempos
if(i==0)
fprintf (pFile, "%s %f\n",str,(float) (tempos[0]-iniciaCaptura));
else
fprintf (pFile, "%s %f\n",str,(float) (tempos[i]-tempos[i-1]));
  }
  fclose (pFile);

  //Video

  //system("cd video");
  (void) sprintf(str, ".\\Resultados\\Video.avi");
  status = CorFileNew(hSystem,str,"w",&hFile);

 // s
 // (void) sprintf(str2, "md Resultados%ld",seconds/3600);
 // system(str2);
  status = CorFileAddSequence(hFile, hFilho, (UINT32) countQuadros , (float) (countQuadros/(TEMPO_DE_AQUISICAO/1000)), " -f avi -c none");

  // Break the connection between acquisition and buffer
  status = CorXferDisconnect(hXfer);

  //printf("Press any key to terminate\n");
 //getchar();

  // Release handles when finished (in the reverse order)
 // status = CorViewFree(hView);
//  status = CorDisplayRelease(hDisplay);
  status = CorXferFree(hXfer);
  status = CorBufferFree(hBuffer);
  status = CorVicFree(hVic);
  status = CorCamFree(hCam);
  status = CorAcqRelease(hAcq);

  for(int i=0;i<NUMERO_DE_QUADROS;i++)
  {
status = CorBufferFree(hFilho[i]);
  }

  // Close Sapera API
  status = CorManClose();

 return 0;
}
```

E Código fonte de captura de imagens 2.0

O programa principal foi dividido em 3 arquivos o Includes.h, Principal.cpp e variaveisGlobais.h

Includes.h

```
#include <stdio.h>

#include <time.h>//biblioteca do comando time

#include <corapi.h>//Biblioteca de captura do framge grabber

#ifdef _CH_ //biblioteca do OpenCv 1.0
#pragma package <opencv>
#endif

#ifndef _EiC
#include "cv.h"
#include "highgui.h"
#endif
#include <iostream>

#define CAPTURA_IMG_VID_MED_POCA 1
#define CAPTURA_IMG_VID_SOMENTE  2
#define CAPTURA_VID_SOMENTE      3
#define VISUALIZA_SOMENTE        4

//#define TEMPO_DE_AQUISICAO 40000 //Tempo em ms
#define NUMERO_DE_QUADROS 6000   //Número máximo de quadros que poderiam ser amostrados
#define TEMANHO_MAXIMO_JANELA 296 //Tamanho máximo da Janela a ser Capturada

////////////////////////////////////////////////////////////////////////
//Novas Funções OpenCV
////////////////////////////////////////////////////////////////////////
#define THRESHOLD_SCAN_LINES 6
#define LINHA_INICIAL_DETECCAO_ARAME 150
#define POSICAO_MEDIA_ARAME 150
#define REGIAO_HORIZONTAL_POSICAO_INICIAL_ARAME 2
#define REGIAO_VERTICAL_A_PARTIR_DAS_BORDAS_INICIAIS_DO_ARAME 50

#define REGIAO_HORIZONTAL_POSICAO_TOQUE_DO_ARAME 10
#define REGIAO_VERTICAL_A_PARTIR_DAS_BORDAS_DO_ARAME 50
```

Principal.cpp

```
#include "Includes.h"
#include "variaveisGlobais.h"


////////////////////////////////////////////////////////////////////
//Funções Para análise de imagens
////////////////////////////////////////////////////////////////////
void FiltroThreshold(IplImage * imagem,int minimo,int maximo,int valorSpofing)
{
for(int i=0;i<imagem->height;i++) //Altura
{
for(int j=0;j<imagem->width;j++) //Largura
{
```

```
if((((uchar*)(imagem->imageData + i*imagem->widthStep))[j] < minimo)||(((uchar*)(imagem->imageData + i*imagem->widthStep))[j] > maximo))
((uchar*)(imagem->imageData + i*imagem->widthStep))[j]= valorSpofing;
}
}
}
//Ok Funciona
void zeraVectorMalloc(int *vetor,int tamanho)
{
for(int i=0;i<tamanho;i++) //Tamanho do vetor
{
vetor[i] = 0;
}

}

//ok Funciona defasada no matlab por 1 ´ja que lá não existe vet[0] mas vet[1]
void somaValorColunasHorizontais(IplImage * imagem,int *vetor,int linhaInicial,int linhaFinal)
{
for(int j=0;j<imagem->width;j++) //Largura
{
for(int i=linhaInicial;i<=linhaFinal;i++) //Altura
{
vetor[j] = vetor[j] + ((uchar*)(imagem->imageData + i*imagem->widthStep))[j];
}
}
}

//ok Funciona defasada no matlab por 1 ´ja que lá não existe vet[0] mas vet[1]
void calculaDerivadaVetor(int tamanhoVetor,int *vetorIn,int *vetorOut)
{
for(int i=0;i<tamanhoVetor-1;i++) //Largura
{
vetorOut[i] = vetorIn[i+1] - vetorIn[i];
}
}


int valorMinimoRegiao(int *vetor,int colunaInicial,int colunaFinal)
{
int menor = 10000; //deve sempre começar com um valor muito alto
int posMenor = 0; //Valor incial em zero
for(int j=colunaInicial;j<colunaFinal;j++)
{
if (vetor[j] < menor)
{
menor = vetor[j];
posMenor = j;
}
}
return posMenor;
}

int valorMaximoRegiao(int *vetor,int colunaInicial,int colunaFinal)
{
int maior = -100000; // deve sempre começar com um valor muito baixo
int posMaior = 0; //Valor incial em zero
for(int j=colunaInicial;j<colunaFinal;j++)
{
if (vetor[j] > maior)
{
maior = vetor[j];
posMaior = j;
}
}
return posMaior;
}
/*
void transformacaoPerspectivaSimples(IplImage * imagemIn,IplImage * imagemOut,int p1x,int p1y,int p2x,int p2y,int p3x,int p3y,int p4x,int p4y)
{
float teta1=0,teta2=0;
int pontoMedioReferencia=0,pontomediaoDesvio=0,tempConta =0;
int deslocamento=0;
pontoMedioReferencia = p1x+(p2x - p1x)/2;
pontomediaoDesvio    = p3x+(p4x - p3x)/2;

printf("pontoMedioReferencia = %i\n",pontoMedioReferencia);
```

```
printf("pontomediaoDesvio = %i\n",pontomediaoDesvio);
// x1   y1  x2  y2 x3   y3  x4 y4
//transformacaoPerspectivaSimples(gray,edge,151,383,337,383,155,167,323,167);
teta1 = atan((double)(p3x-p1x)/(p1y-p3y));
teta2 = atan((double)(p2x-p4x)/(p4y-p2y));

printf("Teta 1 = %f\n",teta1);
printf("Teta 2 = %f\n",teta2);

for(int i=0;i<imagemOut->height;i++) //Altura
{
deslocamento = (p1y - i)*tan(teta1);
printf("%i ",deslocamento);
for(int j=0;j<pontoMedioReferencia;j++) //Largura
{
tempConta = j - deslocamento;
if( tempConta > 0)
((uchar*)(imagemOut->imageData + i*imagemOut->widthStep))[tempConta]= ((uchar*)(imagemIn->imageData + i*imagemIn->widthStep))[j];
}
}

for(int i=0;i<imagemOut->height;i++) //Altura
{
deslocamento = (p2y - i)*tan(teta2);
printf("%i ",deslocamento);
for(int j=pontoMedioReferencia;j<imagemOut->width;j++) //Largura
{
tempConta = j - deslocamento;
if( tempConta > 0)
((uchar*)(imagemOut->imageData + i*imagemOut->widthStep))[tempConta+1]= ((uchar*)(imagemIn->imageData + i*imagemIn->widthStep))[j];
}
}
}
*/
////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////
//Função de análise de Tempo
////////////////////////////////////////////////////////////////////////
//Retorna o tempo decorrido desde o inínico do programa
unsigned long double GetTimeMs()
{
   LARGE_INTEGER m_large_int_tmp_init;
   QueryPerformanceCounter( &m_large_int_tmp_init );
   return ( m_large_int_tmp_init.QuadPart*1000 )/( frequencyPerformancy.QuadPart*1.00 );
}


////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////
//Funções de Exibição do Menu
////////////////////////////////////////////////////////////////////////
void exibeMenuPrincipal()
{
system("cls");
printf("-----------------------------------------------------------------\n");
printf("|\t\t Programa de Aquisicao de Imagens\t\t    |\n");
printf("|                                                  |\n");
printf("|   Entre com a Opcao:                                      |\n");
printf("|                                                  |\n");
printf("|                                                  |\n");
printf("|\t(1)  Capturar imagens, video e medicao de Poca          |\n");
printf("|\t(2)  Somente Capturar imagens e Video                   |\n");
printf("|\t(3)  Somente Capturar Video                             |\n");
printf("|\t(4)  Somente visualizar  Streaming                      |\n");
printf("|\t(5)  Configura Arquivo de configuracao e tempo de Aquisicao |\n");
printf("|\t(0)  Sair                                               |\n");
printf("|                                                  |\n");
printf("-----------------------------------------------------------------\n");
printf("Opcao:");
}

void exibeMenuCapturaImgVidPoca()
{
system("cls");
```

```
printf("-----------------------------------------------------------------\n");
printf("|\t\t Capturar imagens, video e medicao de Poca      |\n");
printf("|                                                    |\n");
printf("|   Entre com a Opcao:                                        |\n");
printf("|                                                    |\n");
printf("|                                                    |\n");
printf("|\t(1)  Alterar arquivo de configurcao de camera           |\n");
printf("|\t     Default ->: %s                 |\n",nomeArquivoCamera);
printf("|\t(2)  Alterar tempo de aquisicao                         |\n");
printf("|\t     Default ->: %i segundos                            |\n",TEMPO_DE_AQUISICAO/1000);
printf("|\t(3)  Voltar Menu anterior                               |\n");
printf("|                                                    |\n");
printf("-----------------------------------------------------------------\n");
printf("Opcao:");
}
//////////////////////////////////////////////////////////////////////

//Função de CallBack: Resposável por todo processamento após a captura de 1 frame
CORSTATUS CCONV XferCallback (void *context, UINT32 eventType, UINT32 eventCount)
{

 if(TipoComandoMenu == VISUALIZA_SOMENTE)
 {
status = CorViewNew(hSystem, hDisplay, hBuffer, CORVIEW_VAL_MODE_AUTO_DETECT,&hView);
CorViewShow(hView);
 }
 else
 {
if(TipoComandoMenu != CAPTURA_IMG_VID_MED_POCA) //Captura imagens e video
{
//Transfere o frame capturado para o Formato OpenCV
status = CorBufferRead(hBuffer, 0, gray->imageData, size);

//Retorna o frame processado novemnte para o Formato Sapera
status = CorBufferWrite(hFilho[countQuadros], 0, gray->imageData, size);

    // Status = CorViewNew(hSystem, hDisplay, hBuffer, CORVIEW_VAL_MODE_AUTO_DETECT,&hView);

// CorViewShow(hView);

}
else //Captura imagens e video e mede poca
{

//zera Vetores de porcessamento
zeraVectorMalloc(VetorSomaLinhasVerticias,gray->width);
zeraVectorMalloc(VetorDerivada1LinhasVerticais,gray->width);

zeraVectorMalloc(VetorTempDiffVerticalPontoMedioArame,gray->height);
zeraVectorMalloc(VetorDiffVerticalPontoMedioArame,gray->height);


//Transfere o frame capturado para o Formato OpenCV
status = CorBufferRead(hBuffer, 0, gray->imageData, size);

//Filtra ScanLines
FiltroThreshold(gray,THRESHOLD_SCAN_LINES,255,0);

//Filtro de Mediana para suavizar picos
cvSmooth(gray, gray,CV_MEDIAN,3);

//gera o primeiro vetorsoma  para identificar a posição das bordas do arame
somaValorColunasHorizontais(gray,VetorSomaLinhasVerticias,
                                LINHA_INICIAL_DETECCAO_ARAME-REGIAO_HORIZONTAL_POSICAO_INICIAL_ARAME,
                                LINHA_INICIAL_DETECCAO_ARAME+REGIAO_HORIZONTAL_POSICAO_INICIAL_ARAME);

//Calcula as derivadas afim de encontrar essas os picos das derivadas
calculaDerivadaVetor(gray->width,VetorSomaLinhasVerticias,VetorDerivada1LinhasVerticais);

//Encontra o inicio da borda do arame encontrando o valor de mínimo restrito a uma regiao
vetorAfericoes[countQuadros].posicaoInicialArame = valorMinimoRegiao(VetorDerivada1LinhasVerticais,
POSICAO_MEDIA_ARAME-REGIAO_VERTICAL_A_PARTIR_DAS_BORDAS_INICIAIS_DO_ARAME,
POSICAO_MEDIA_ARAME+REGIAO_VERTICAL_A_PARTIR_DAS_BORDAS_INICIAIS_DO_ARAME);

//Encontra o Fim da borda do arame encontrando o valor de máximo restrito a uma regiao
vetorAfericoes[countQuadros].posicaoFinalArame = valorMaximoRegiao(VetorDerivada1LinhasVerticais,
```

```
POSICAO_MEDIA_ARAME-REGIAO_VERTICAL_A_PARTIR_DAS_BORDAS_INICIAIS_DO_ARAME,
POSICAO_MEDIA_ARAME+REGIAO_VERTICAL_A_PARTIR_DAS_BORDAS_INICIAIS_DO_ARAME);


//Copia a linha vertical do arame para um vetor a partir do ponto medio das bordas do arame
posicaoMediaArame = (vetorAfericoes[countQuadros].posicaoFinalArame -
 vetorAfericoes[countQuadros].posicaoInicialArame)/2 +
 vetorAfericoes[countQuadros].posicaoInicialArame;
//printf("\n O medio valor e: %i\n",posicaoMediaArame);

for(int j=0; j<gray->height;j++)
{
VetorTempDiffVerticalPontoMedioArame[j] = ((uchar*)(gray->imageData + j*gray->widthStep))[posicaoMediaArame];
}


//calcula a derivada da linha vertical afim de encontrar o ponto de toque do arame
calculaDerivadaVetor(gray->height,VetorTempDiffVerticalPontoMedioArame,VetorDiffVerticalPontoMedioArame);

//encontro o valor de máximo ou seja o ponto de toque do arame na poça
vetorAfericoes[countQuadros].posicaotoquearame = valorMaximoRegiao(VetorDiffVerticalPontoMedioArame,0,gray->height);

//gera o segundo vetorsoma  para identificar a posição das bordas do arame
zeraVectorMalloc(VetorSomaLinhasVerticias,gray->width);
zeraVectorMalloc(VetorDerivada1LinhasVerticais,gray->width);

somaValorColunasHorizontais(gray,VetorSomaLinhasVerticias,
vetorAfericoes[countQuadros].posicaotoquearame-REGIAO_HORIZONTAL_POSICAO_TOQUE_DO_ARAME,
vetorAfericoes[countQuadros].posicaotoquearame+REGIAO_HORIZONTAL_POSICAO_TOQUE_DO_ARAME);

//calcula a derivada da linha horizontal afim de encontrar as bordas da poça e arame
calculaDerivadaVetor(gray->width,VetorSomaLinhasVerticias,VetorDerivada1LinhasVerticais);

//Encontra o inicio da borda da poça encontrando o valor de máximo restrito a uma regiao
vetorAfericoes[countQuadros].posicaoInicialBorda = valorMaximoRegiao(VetorDerivada1LinhasVerticais,
0,vetorAfericoes[countQuadros].posicaoInicialArame - REGIAO_VERTICAL_A_PARTIR_DAS_BORDAS_DO_ARAME);

//Encontra o Fim da borda da poça encontrando o valor de mínimo restrito a uma regiao
vetorAfericoes[countQuadros].posicaoFinalBorda = valorMinimoRegiao(VetorDerivada1LinhasVerticais,
vetorAfericoes[countQuadros].posicaoFinalArame + REGIAO_VERTICAL_A_PARTIR_DAS_BORDAS_DO_ARAME,
gray->width);

//Retorna o frame processado novemnte para o Formato Sapera
status = CorBufferWrite(hFilho[countQuadros], 0, gray->imageData, size);



}
//Grava o tempo decorrido entr frames em em un vetor de tempos
tempos[countQuadros] = GetTimeMs() - iniciaCaptura;
//Exibe imagem em uma tela não gerenciada
// status = CorViewNew(hSystem, hDisplay, hFilho[countQuadros], CORVIEW_VAL_MODE_AUTO_DETECT,
//   &hView);

// CorViewShow(hView);

 }

//Incrementa contador de frames
countQuadros++;

    return CORSTATUS_OK;
}


int main()
{
  while(menuPrincipal !='z')
  {
  exibeMenuPrincipal();
  std::cin>>menuPrincipal;
  switch(menuPrincipal)
  {
case '1':
{
system("cls");
```

```
printf("Iniciando a Captura... ");
TipoComandoMenu = CAPTURA_IMG_VID_MED_POCA;
menuPrincipal ='z';
}
break;

case '2':
system("cls");
printf("Iniciando a Captura... ");
TipoComandoMenu = CAPTURA_IMG_VID_SOMENTE;
menuPrincipal ='z';
break;

case '3':
system("cls");
printf("Iniciando a Captura... ");
TipoComandoMenu = CAPTURA_VID_SOMENTE;
menuPrincipal ='z';
break;

case '4':
system("cls");
printf("Iniciando a Captura... ");
TipoComandoMenu = VISUALIZA_SOMENTE;
menuPrincipal ='z';
break;

case '5':
exibeMenuCapturaImgVidPoca();
std::cin>>menuCapturaImgVidPoca;
switch(menuCapturaImgVidPoca)
{
case '1':
system("cls");
printf("Entre com o novo nome do arquivo de configuracao: ");
scanf("%s",nomeArquivoCamera);
printf("\nNome do arquivo alterado para: %s\n",nomeArquivoCamera);
Sleep(3000);
break;

case '2':
system("cls");
printf("Entre com o novo Tempo de aquisicao (em segundos)\n");
printf(" Max 60s  a 150 fps 128X128 pixels\n ");
printf(" Max 69s  a 130 fps 128X128 pixels\n ");
printf(" Max 75s  a 120 fps 128X128 pixels\n ");
printf(" Max 90s  a 100 fps 128X128 pixels\n ");
printf(" Max 150s a 60  fps 128X128 pixels\n ");
printf(" Max 225s a 40  fps 128X128 pixels\n ");
printf(" Max 450s a 20  fps 128X128 pixels\n\n ");
printf("Tempo:");
std::cin>>TEMPO_DE_AQUISICAO;
TEMPO_DE_AQUISICAO = TEMPO_DE_AQUISICAO * 1000;
printf("\nTempo de aquisicao alterado para: %i\n segundos",TEMPO_DE_AQUISICAO/1000);
Sleep(3000);
break;

case '3':
menuCapturaImgVidPoca=9;
break;

default:
printf("Opcao Invalida");
Sleep(1000);
break;
}
break;

case '0':
return 0;
break;

default:
printf("Opcao Invalida");
Sleep(1000);
break;
```

```
  }
  }

  ////////////////////////////////////////////////////////////////////
  //Configura o Timer
  ////////////////////////////////////////////////////////////////////
  //Captura quantos Ticks por segundo
  QueryPerformanceFrequency( &frequencyPerformancy );
  ////////////////////////////////////////////////////////////////////

  ////////////////////////////////////////////////////////////////////
  //Configura o Frame Grabber
  ////////////////////////////////////////////////////////////////////
  // Initialize Sapera API
  status = CorManOpen();

  // Get server handles (system and board)
  hSystem = CorManGetLocalServer();
  status = CorManGetServerByName("X64-CL_iPro_1", &hBoard);

  // Get acquisition handle
  status = CorAcqGetHandle(hBoard, 0, &hAcq);  // 0 = First instance

  // Create CAM/VIC handles
  status = CorCamNew(hSystem, &hCam);  // Camera
  status = CorVicNew(hSystem, &hVic);  // Video-Input-Conditionning

  // Load CAM/VIC parameters from file into system memory
  // The acquisition hardware is not initialized at this point
  status = CorCamLoad(hCam, nomeArquivoCamera);
  status = CorVicLoad( hVic, nomeArquivoCamera);

  // Download the CAM/VIC parameters to the acquisition module
  // The acquisition hardware is now initialized
  status = CorAcqSetPrms(hAcq, hVic, hCam, FALSE);

  // Create a buffer compatible to acquisition
  status = CorAcqGetPrm(hAcq, CORACQ_PRM_SCALE_HORZ, &width);
  status = CorAcqGetPrm(hAcq, CORACQ_PRM_SCALE_VERT, &height);
  status = CorAcqGetPrm(hAcq, CORACQ_PRM_OUTPUT_FORMAT, &format);
  status = CorBufferNew(hSystem, width, height, format,
    CORBUFFER_VAL_TYPE_CONTIGUOUS, &hBuffer);

  for(int i=0;i<NUMERO_DE_QUADROS;i++)
  {
status = CorBufferNew(hSystem, width, height, format,
    CORBUFFER_VAL_TYPE_SCATTER_GATHER,&hFilho[i]);
 status = CorXferNew(hBoard, hAcq, hFilho[i], NULL, &hXfer);
  }

  // Create a transfer handle to link acquisition to buffer
  status = CorXferNew(hBoard, hAcq, hBuffer, NULL, &hXfer);

  // Register a callback function on "End-Of-Frame" events
  status = CorXferRegisterCallback(hXfer, CORXFER_VAL_EVENT_TYPE_END_OF_FRAME,
      XferCallback, (void *)&hBuffer);

  // Activate the connection between acquisition and buffer
  status = CorXferConnect(hXfer);

  // Get display handle
  status = CorDisplayGetHandle(hSystem, 0, &hDisplay);

  ////////////////////////////////////////////////////////////////////

  ////////////////////////////////////////////////////////////////////
  //Configura Imagens de TRabalho para o OpenCV
  ////////////////////////////////////////////////////////////////////
  //cria buffer de imagem em format OpenCV
  gray = cvCreateImage(cvSize(width,height), IPL_DEPTH_8U, 1);
 // edge = cvCreateImage(cvSize(width,height), IPL_DEPTH_8U, 1);

  //define o tamanho da imagem total
  size = width*height * sizeof( UINT8);
  ////////////////////////////////////////////////////////////////////
```

```
  ////////////////////////////////////////////////////////////////////
  //IniCia a Captura
  ////////////////////////////////////////////////////////////////////
  printf("Dispositivo Pronto Digite 0 para Continuar:");
  std::cin>>menuCapturaImgVidPoca;
  printf("\nCapturando...\n");
  // Start a continuous transfer (live grab)
  status = CorXferStart(hXfer, CORXFER_CONTINUOUS);
  if(TipoComandoMenu == VISUALIZA_SOMENTE)
  {
printf("\nDigite 1 Para terminar:");
int tempvar =0;
while(tempvar==0)
std::cin>>tempvar;
  }
  else
  {
  iniciaCaptura = GetTimeMs();
  Sleep(TEMPO_DE_AQUISICAO);
  }

  printf("\nFim Captura...");
  status = CorXferStop(hXfer);
  //status = CorXferWait(hXfer, 1000);
  ////////////////////////////////////////////////////////////////////

  if(TipoComandoMenu == VISUALIZA_SOMENTE)
  {
  printf("\nFechado programa...\n");
  }
  else
  {
  ////////////////////////////////////////////////////////////////////
  //Salva Figuras / Video / Tempos em arquivo
  ////////////////////////////////////////////////////////////////////
  printf("\nCriando Arquivos...\n");
  char str [80],str2[80];// String que cria arquivo
  seconds = time (NULL);
 // (void) sprintf(str2, "md Resultados%ld", seconds);
 // system(str2);
  system("md Resultados");
  pFile = fopen (".\\Resultados\\Tempos.txt","w");

  if(TipoComandoMenu == CAPTURA_IMG_VID_MED_POCA)
  {
pFile2 = fopen (".\\Resultados\\Medidas.txt","w");
//printf("teste");
  }
  for(int i=0;i<countQuadros;i++)
  {
(void) sprintf(str, ".\\Resultados\\Img%i.bmp",i);

if(TipoComandoMenu !=CAPTURA_VID_SOMENTE)//pode capturar img e medidas
{
//Forma mais Atual
status = CorFileNew(hSystem,str,"wb",&hFile);
status = CorFileSave(hFile, (CORHANDLE) hFilho[i],"-f corfile_format_bmp");

//Forma Obsoleta
//status = CorBufferSave(hFilho[i], str, CORFILE_VAL_FORMAT_BMP);
 if(TipoComandoMenu == CAPTURA_IMG_VID_MED_POCA)
 {
fprintf (pFile2, "Img%i.bmp ",i);
fprintf (pFile2, "posicaoInicialArame %i ",vetorAfericoes[i].posicaoInicialArame);
fprintf (pFile2, "posicaoFinalArame %i ",vetorAfericoes[i].posicaoFinalArame);
fprintf (pFile2, "posicaoInicialBorda %i ",vetorAfericoes[i].posicaoInicialBorda);
fprintf (pFile2, "posicaoFinalBorda  %i ",vetorAfericoes[i].posicaoFinalBorda);
fprintf (pFile2, "posicaotoquearame  %i \n",vetorAfericoes[i].posicaotoquearame);

 }
}
(void) sprintf(str, "Img%i.bmp",i);
//Escreve tempos
if(i==0)
fprintf (pFile, "%s %f \n",str,(float) (tempos[0]-iniciaCaptura));
```

```
else
fprintf (pFile, "%s %f \n",str,(float) (tempos[i]-tempos[i-1]));
  }
  //Video
  (void) sprintf(str, ".\\Resultados\\Video.avi");
  status = CorFileNew(hSystem,str,"w",&hFile);
  status = CorFileAddSequence(hFile, hFilho, (UINT32) countQuadros , (float) (countQuadros/(TEMPO_DE_AQUISICAO/1000)), " -f avi -c none");

  ////////////////////////////////////////////////////////////////////
  printf("\nFechado programa...\n");
  }

  ////////////////////////////////////////////////////////////////////
  //Fecha processo e Destroi variáveis Dinamicas
  ////////////////////////////////////////////////////////////////////

  // Break the connection between acquisition and buffer
  status = CorXferDisconnect(hXfer);

  // Release handles when finished (in the reverse order)
  status = CorViewFree(hView);
  status = CorDisplayRelease(hDisplay);
  status = CorXferFree(hXfer);
  status = CorBufferFree(hBuffer);
  status = CorVicFree(hVic);
  status = CorCamFree(hCam);
  status = CorAcqRelease(hAcq);

  for(int i=0;i<NUMERO_DE_QUADROS;i++)
  {
status = CorBufferFree(hFilho[i]);
  }

  // Close Sapera API
  status = CorManClose();

  //Libera variáveis OpenCV
  cvReleaseImage(&image);
  cvReleaseImage(&gray);
  cvReleaseImage(&edge);

 system("pause");
 fclose (pFile);
 fclose (pFile2);
 return 0;
 ////////////////////////////////////////////////////////////////////
}
```

variaveisGlobais.h

```
////////////////////////////////////////////////////////////////////
//VARIÁVEIS RELACIONADAS COM API DO FRAME GRABBER
////////////////////////////////////////////////////////////////////

//Base para a Captura
CORSERVER hBoard;     // Board server  handle
CORCAM hCam;          // CAM           handle
CORVIC hVic;          // VIC           handle
CORACQ hAcq;          // Acquisition   handle
CORXFER hXfer;        // Transfer      handle
CORFILE hFile;        // File      handle
CORSTATUS status;     // Error code
CORSERVER hSystem;    // System server handle

//Dados de Captura
CORBUFFER hBuffer; // Buffer handle
CORBUFFER hFilho[NUMERO_DE_QUADROS];    // Buffer Filho on serao armazenados os quadros
UINT32 size; // Size of the image in bytes
int countQuadros =0; //Contador de quadros

//Dados de visualização
CORVIEW hView;        // View handle
```

```
CORDISPLAY hDisplay;  // Display handle
UINT32 width, height, format;

//////////////////////////////////////////////////////////////////////
//Dados da Biblioteca OpenCV
//////////////////////////////////////////////////////////////////////

IplImage *image = 0, *gray = 0, *edge = 0;

//////////////////////////////////////////////////////////////////////
//Dados da Biblioteca de aferição de tempo
//////////////////////////////////////////////////////////////////////

LARGE_INTEGER frequencyPerformancy;  //Recebe qunatos ticks o sistema tem por segundo
unsigned long double iniciaCaptura=0,fimCaptura=0;
unsigned long double tempos[NUMERO_DE_QUADROS];

//////////////////////////////////////////////////////////////////////
//Dados de Medição da poça de soldagem
//////////////////////////////////////////////////////////////////////
struct Afericoes {
  int posicaoInicialArame;
  int posicaoFinalArame;
  int posicaoInicialBorda;
  int posicaoFinalBorda;
  int posicaotoquearame;
};

Afericoes vetorAfericoes[NUMERO_DE_QUADROS];

int VetorSomaLinhasVerticias[TEMANHO_MAXIMO_JANELA];
int VetorDiffVerticalPontoMedioArame[TEMANHO_MAXIMO_JANELA];
int VetorDerivada1LinhasVerticais[TEMANHO_MAXIMO_JANELA];
int VetorTempDiffVerticalPontoMedioArame[TEMANHO_MAXIMO_JANELA];

int posicaoMediaArame = 0;
int posicaoDeToqueDoArameNaPoca = 0;
//////////////////////////////////////////////////////////////////////
//Dados gravação de arquivos em disco
//////////////////////////////////////////////////////////////////////
FILE * pFile, *pFile2;
time_t seconds;

//////////////////////////////////////////////////////////////////////
//Dados de configuração do experimento
//////////////////////////////////////////////////////////////////////
//char nomeArquivoCamera[80]= "AjusteRobo128_128.ccf"; //Arquivo de configuração da camera
//char nomeArquivoCamera[80]= "fullRes.ccf"; //Arquivo de configuração da camera
//char nomeArquivoCamera[80]= "Experimento256_256.ccf";
//char nomeArquivoCamera[80]= "janela_300_300.ccf";
char nomeArquivoCamera[80]= "janela300_296.ccf";

int TEMPO_DE_AQUISICAO = 2000 ;//Tempo em ms

char menuPrincipal=0,menuCapturaImgVidPoca=0; //Variáveis do menu
int TipoComandoMenu=0; //Tipo de comando a ser realizado
```

F Programa de testes de imagem offline em MatLab 1.0

```
clear all
clc
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Abre Imagem
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
I=(imread('figTese04.bmp'));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%filtro threshold para retirar scanlines
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
a = I>20;
c = cast(a,'uint8');
d = I.*c;
%imview(d);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%soma de brilho Total
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
brilhoTotal = sum(sum(d));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Calculo da Media Total sem caracteres zero
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
brilhoMedio = brilhoTotal/sum(sum(d>0));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Exibe resultado do filtro de Brilho Médio
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
bm = I.*cast((d>brilhoMedio),'uint8');
imview(bm);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Calculo da intensidade de brilho das colunas
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
vetorSomaLinhasVerticais = sum(d);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Filtro de valores maiores que a média
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
e = (vetorSomaLinhasVerticais>brilhoMedio);
vetorSomaLinhasVerticais= vetorSomaLinhasVerticais .* e;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Aplicar a Primeira derivada sobre o vetor vetorSomaLinhasVerticais
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
vetorDerivadaLinhasVerticais = diff(vetorSomaLinhasVerticais);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Aplicar a Segunda derivada sobre o vetor vetorSomaLinhasVerticais
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
vetorDerivada2LinhasVerticais = diff(vetorDerivadaLinhasVerticais);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Calcula a posição inicial e final da Borda sa poça de soldagem
% sobre o vetor vetorDerivadaLinhasVerticais
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[ValorLixo,inicioBorda] = max(vetorDerivadaLinhasVerticais);
[ValorLixo,fimBorda] = min(vetorDerivadaLinhasVerticais);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Calcula a posição do arame no vetorDerivada2LinhasVerticais baseado na
%regia definida por inicioBorda e fimBorda
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
desvio = 3; %valor que pode variar o intervalo para mais ou para menos
maior = -100000; % deve sempre começar com um valor muito baixo
menor = 100000; %deve sempre começar com um valor muito alto
posMaior = 0; %Valor incial em zero
posMenor = 0; %Valor incial em zero
%varre o vetor vetorDerivada2LinhasVerticais em uma regiao delimitada por
%inicioBorda e fimBorda
for m = (inicioBorda+desvio):(fimBorda-desvio)
    %detecta o maior valor e posição
    if (vetorDerivada2LinhasVerticais(m) > maior)
        maior = vetorDerivada2LinhasVerticais(m);
        posMaior = m;
    end
    %detecta o menor valor e posição
    if (vetorDerivada2LinhasVerticais(m) < menor)
        menor = vetorDerivada2LinhasVerticais(m);
        posMenor = m;
    end
end
%Calcula o ponto médio entre a posição do  maior e menor valor
posArame = posMaior+cast((posMenor - posMaior)/2,'uint8');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Imprime REsultados
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%imprime a figura original
%  figure;
%  A = sum(I);
%  plot(A);

%imprime o resultado do filtro de ruido de fundo
% figure;
% A = sum(d);
% plot(A);

%imprime o resultado do filtro de valores acima da média
% figure;
% plot(vetorSomaLinhasVerticais);

%imprime a derivada do vetor vetorSomaLinhasVerticais
%figure;
%plot(vetorDerivadaLinhasVerticais);

%imprime a derivada do vetor vetorSomaLinhasVerticais
%figure;
%plot(vetorDerivada2LinhasVerticais);

%imprime resultados
inicioBorda
fimBorda
tamahoPoca = fimBorda - inicioBorda
posArame
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

G Programa de testes de imagem offline em MatLab 2.0

```
clear all
clc
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Abre Imagem
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
I=(imread('Img525.bmp'));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%imview(I)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Filtra a imagem Retira as ScanLines de Ruido
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
a = I>6;
c = cast(a,'uint8');
I = I.*c;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%imview(I)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Filtra a imagem com uum filtro tipo mediana 3X3
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
H = fspecial('average',3);
I= imfilter(I,H,'replicate');
imview(I);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Escolhe um Valor inicial da posição do Arame
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
valorChuteLinhaInicial = 150;
tamanhoRegiaoChuteLinha = 2;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Deriva a Linha Horizontal indicada pela melho posicao
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
vetorSomaLinhasVerticais = diff(sum(I(valorChuteLinhaInicial-tamanhoRegiaoChuteLinha:valorChuteLinhaInicial+tamanhoRegiaoChuteLinha,:)));
figure;
plot(vetorSomaLinhasVerticais);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Encontra as Bordas refewrentes ao inicio e Fim da borda Arame
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[valor,posicaInicioBordaArame] = min(vetorSomaLinhasVerticais(150-50:150+50));
[valor,posicaFimBordaArame] = max(vetorSomaLinhasVerticais(150-50:150+50));
posicaInicioBordaArame = posicaInicioBordaArame + 150-50;
posicaFimBordaArame = posicaFimBordaArame + 150-50;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Encontra Posição Média do Arame
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
posicaoMediaArame = (posicaFimBordaArame - posicaInicioBordaArame)/2 + posicaInicioBordaArame;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Encontra Ponto de Toque Do Arame provável
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
vetorDiff1PontoMedioArame = diff(I(:,ceil(posicaoMediaArame)));
figure;
plot(vetorDiff1PontoMedioArame);

[valor,posMelhorLinhaBorda] = max(vetorDiff1PontoMedioArame);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Encontra Bordas Da Poça de soldagem
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
vetorSomaLinhasVerticais = diff(sum(I(posMelhorLinhaBorda-tamanhoRegiaoChuteLinha:posMelhorLinhaBorda+tamanhoRegiaoChuteLinha,:)));
figure;
plot(vetorSomaLinhasVerticais);

% vetorSomaLinhasVerticais = diff(I(posMelhorLinhaBorda,:));
% figure;
% plot(vetorSomaLinhasVerticais);

[valor,posicaInicioBorda] = max(vetorSomaLinhasVerticais(1:posicaInicioBordaArame-10));
[valor,posicaFimBorda] = min(vetorSomaLinhasVerticais(posicaFimBordaArame+10:length(vetorSomaLinhasVerticais)));

posicaFimBorda = posicaFimBorda + posicaFimBordaArame +10;


posicaInicioBorda
posicaFimBorda

posicaInicioBordaArame
posicaFimBordaArame
```

H Programa analise de imagens offline 1.0

```
#ifdef _CH_
#pragma package <opencv>
#endif
#include <cstdlib>
#include "stdio.h"
#include <cstdlib>
#include <iostream>
#ifndef _EiC
#include "cv.h"
#include "highgui.h"
#endif


using namespace std;

char wndname[] = "Edge";
char tbarname[] = "Threshold";
int edge_thresh = 1;

int vectorMedioFRame[1200];

IplImage *image = 0,*gray = 0;

struct medidas {
  int brilhoTotal;
  int brilhoMedio;
  int iniPoca;
  int fimPoca;
  int tamPoca;
  int posArame;
};



//ok Funciona
int brilhoTotalImagem(IplImage * imagem)
{
int soma=0;
for(int i=0;i<imagem->height;i++) //Altura
{
for(int j=0;j<imagem->width;j++) //Largura
{
soma = soma + ((uchar*)(imagem->imageData + i*imagem->widthStep))[j];
}
}
return soma;
}
//Ok Funciona
int calculaMediaFigura(IplImage * imagem, int brilhoTotalImagem)
{
return brilhoTotalImagem/(imagem->width*imagem->height);
}
//ok Funciona
void FiltroThreshold(IplImage * imagem,int minimo,int maximo,int valorSpofing)
{
for(int i=0;i<imagem->height;i++) //Altura
{
for(int j=0;j<imagem->width;j++) //Largura
{
if((((uchar*)(imagem->imageData + i*imagem->widthStep))[j] < minimo)||(((uchar*)(imagem->imageData + i*imagem->widthStep))[j] > maximo))
((uchar*)(imagem->imageData + i*imagem->widthStep))[j]= valorSpofing;
}
}
}
int contaPixelsDiferenteDeZero(IplImage * imagem)
{
int soma=0;
```

```
for(int i=0;i<imagem->height;i++) //Altura
{
for(int j=0;j<imagem->width;j++) //Largura
{
if((((uchar*)(imagem->imageData + i*imagem->widthStep))[j] > 0))
soma++;
}
}
return soma;
}
//Ok Funciona
void zeraVectorMalloc(int *vetor,int tamanho)
{
for(int i=0;i<tamanho;i++) //Tamanho do vetor
{
vetor[i] = 0;
}

}

//ok Funciona defasada no matlab por 1 ´ja que lá não existe vet[0] mas vet[1]
void somaValorColunasHorizontais(IplImage * imagem,int *vetor)
{
for(int j=0;j<imagem->width;j++) //Largura
{
for(int i=0;i<imagem->height;i++) //Altura
{
vetor[j] = vetor[j] + ((uchar*)(imagem->imageData + i*imagem->widthStep))[j];
}
}
}

//ok Funciona defasada no matlab por 1 ´ja que lá não existe vet[0] mas vet[1]
void calculaDerivadaVetor(IplImage * imagem,int *vetorIn,int *vetorOut)
{
for(int i=0;i<imagem->width-1;i++) //Largura
{
vetorOut[i] = vetorIn[i+1] - vetorIn[i];
}
}
//ok Funciona
void transformacaoPerspectivaSimples(IplImage * imagemIn,IplImage * imagemOut,int p1x,int p1y,int p2x,int p2y,int p3x,int p3y,int p4x,int p4y)
{
float teta1=0,teta2=0;
int pontoMedioReferencia=0,pontomediaoDesvio=0,tempConta =0;
int deslocamento=0;
pontoMedioReferencia = p1x+(p2x - p1x)/2;
pontomediaoDesvio    = p3x+(p4x - p3x)/2;

printf("pontoMedioReferencia = %i\n",pontoMedioReferencia);
printf("pontomediaoDesvio = %i\n",pontomediaoDesvio);
// x1   y1  x2  y2 x3   y3  x4 y4
//transformacaoPerspectivaSimples(gray,edge,151,383,337,383,155,167,323,167);
teta1 = atan((double)(p3x-p1x)/(p1y-p3y));
teta2 = atan((double)(p2x-p4x)/(p4y-p2y));

printf("Teta 1 = %f\n",teta1);
printf("Teta 2 = %f\n",teta2);

for(int i=0;i<imagemOut->height;i++) //Altura
{
deslocamento = (p1y - i)*tan(teta1);
printf("%i ",deslocamento);
for(int j=0;j<pontoMedioReferencia;j++) //Largura
{
tempConta = j - deslocamento;
if( tempConta > 0)
((uchar*)(imagemOut->imageData + i*imagemOut->widthStep))[tempConta]= ((uchar*)(imagemIn->imageData + i*imagemIn->widthStep))[j];
}
}

for(int i=0;i<imagemOut->height;i++) //Altura
{
deslocamento = (p2y - i)*tan(teta2);
printf("%i ",deslocamento);
```

```
for(int j=pontoMedioReferencia;j<imagemOut->width;j++) //Largura
{
tempConta = j - deslocamento;
if( tempConta > 0)
((uchar*)(imagemOut->imageData + i*imagemOut->widthStep))[tempConta+1]= ((uchar*)(imagemIn->imageData + i*imagemIn->widthStep))[j];
}
}
}
////////////////////////////////////////////////////////////////////////

//ok Funciona
void imprimeVetor(IplImage * imagem,int *vetorIn)
{
for (int i=0;i<imagem->width;i++)
printf("%i ",vetorIn[i]);

printf("\n\n");
}
////////////////////////////////////////////////////////////////////////

//ok Funciona
void detectaMaiorMenor(int *vetorIn,int &Maior,int &posMaior,int &Menor, int &posMenor, int posInicio,int posFim,int desvio)
{
Maior = -100000; // deve sempre começar com um valor muito baixo
Menor = 100000; //deve sempre começar com um valor muito alto
posMaior = 0; //Valor incial em zero
posMenor = 0; //Valor incial em zero

//varre o vetor vetorDerivada2LinhasVerticais em uma regiao delimitada por
//inicioBorda e fimBorda
for(int i=(posInicio + desvio);i <(posFim - desvio); i++)
{   //detecta o maior valor e posição
if(vetorIn[i] > Maior)
{
Maior = vetorIn[i];
posMaior = i;
}
if (vetorIn[i] < Menor)
{
Menor = vetorIn[i];
posMenor = i;
}
}
}
////////////////////////////////////////////////////////////////////////

void on_trackbar(int h)
{
    cvShowImage(wndname, gray);
}

void exibeMenuPrincipal()
{
system("cls");
printf("-------------------------------------------------------------------------\n");
printf("|\t\t Programa de Analise de Imagens OFF Line\t    |\n");
printf("|                                                         |\n");
printf("|   Entre com a Opcao:                                                |\n");
printf("|                                                         |\n");
printf("|                                                         |\n");
printf("|\t(1)  Definir o numeros de Arquivos a serem processados      |\n");
printf("|\t(2)  Alterar o range de Brilho máximo e mínimo de uma poça  |\n");
printf("|\t(5)  Processa arquivos                                      |\n");
printf("|\t(0)  Sair                                                   |\n");
printf("|                                                         |\n");
printf("-------------------------------------------------------------------------\n");
printf("Opcao:");
}
int main( int argc, char** argv )
{
  char menuPrincipal = 0;

  int posArquivoInicial = 0;
  int posArquivoFinal = 1;

  int brilhoMaxPoca = 1000000;
```

```
  int brilhoMinPoca = 0;

  while(menuPrincipal !='z')
  {
  exibeMenuPrincipal();
  std::cin>>menuPrincipal;
  switch(menuPrincipal)
  {
case '1':
{
system("cls");
printf("Varredura Default de img%i.bmp ate img%i.bmp",posArquivoInicial,posArquivoFinal);
printf("\nEntre com o valor do arquivo Inicial: ");
std::cin>>posArquivoInicial;
printf("\nEntre com o valor do arquivo Final: ");
std::cin>>posArquivoFinal;
printf("\nVarredura alterada para img%i.bmp ate img%i.bmp",posArquivoInicial,posArquivoFinal);
//Sleep(3000);
}
break;

case '2':
system("cls");
printf("Intervalo de Brilho entre %i <-> %i",brilhoMinPoca,brilhoMaxPoca);
printf("\nEntre com o valor do Brilho minimo: ");
std::cin>>brilhoMinPoca;
printf("\nEntre com o valor do Brilho maximo: ");
std::cin>>brilhoMaxPoca;
printf("\nIntervalo de Brilho Alterdo para %i <-> %i",brilhoMinPoca,brilhoMaxPoca);
//sleep(3000);
break;

case '5':
{
system("cls");
printf("Processando arquivos...\n");
menuPrincipal ='z';
}
break;

case '0':
return 0;
break;

default:
printf("Opcao Invalida");
//Sleep(1000);
break;
  }
  }
int Maior = 0,posMaior = 0,Menor = 0,posMenor = 0;

//Cria dinamicamente Vetores para 1 somas e 2 derivadas das colunas horizontais.
medidas *vetorMedidas;
//Malloc Vetores de armazenamento das varávies de medição
vetorMedidas = (medidas *) calloc(posArquivoFinal+1,sizeof(medidas));
if (vetorMedidas==NULL)
printf("Falha no Malloc vetorSomaLinhasVerticais\n");

char str [80]; //Cria Vetor para escrever Strings
FILE * pFile;  //Ponterio para escrever em arquivos

//Cria uma janela no windows chamada "wndname"
    cvNamedWindow(wndname, 1);
pFile = fopen (".\\res\\AnaliseExperimento.txt","w");
//int i=0; //Desvio caso queira analizar um só aqruivo imgX.bmp
//Controle de Loop em arquivos do tipo "imgX.bmp" img1.bmp,img2.bmp ... img2704.bmp
for (int i=posArquivoInicial;i<posArquivoFinal;i++)
{
//Cria a string para carregar os arquivos localizados no diretório \org
(void) sprintf(str,".\\org\\img%i.bmp", i);
//Carrega o arquivo em uma Variavel de imagem ou sai do programa de uma vez
if( (image = cvLoadImage( str, 1)) == 0 )
        return -1;
//Cria o arquivo de resultados no diretorio
```

```
//Aloca a imagem em uma variável do tipo imagem
gray = cvCreateImage(cvSize(image->width,image->height), IPL_DEPTH_8U, 1);
    // edge = cvCreateImage(cvSize(image->width,image->height), IPL_DEPTH_8U, 1);
    cvCvtColor(image, gray, CV_BGR2GRAY); //Força a conversão da imagem em grayscale

//Filtra o ruido de fundo
FiltroThreshold(gray,20  ,255,0); //Filtra ScanLines

//Calcula  o brilho total da imagem
vetorMedidas[i].brilhoTotal = brilhoTotalImagem(gray);

//Se o brilho estiver em uma região determinada  deve-se então analisar a imagem
    if((vetorMedidas[i].brilhoTotal>brilhoMinPoca)&&(vetorMedidas[i].brilhoTotal<brilhoMaxPoca))
{

//Cria a string para grava no arquivo de resultados
fprintf (pFile, "img%i Brilho total: %i P ",i,vetorMedidas[i].brilhoTotal);

//Copia o arquivo processado em uma pasta diferente
//dessa forma
//(void) sprintf(str,"copy .\\vid\\img%i.bmp .\\res\\", i);
//system(str);
//OU dessa forma
(void) sprintf(str,".\\res\\img%i.bmp", i);
cvSaveImage(str,gray);

//Cria dinamicamente Vetores para 1 somas e 2 derivadas das colunas horizontais.
int *vetorSomaLinhasVerticais,*vetorDerivadaLinhasVerticais,*vetorDerivada2LinhasVerticais;
//Malloc Vetores de armazenamento das varávies de medição
vetorSomaLinhasVerticais = (int *) calloc((int)gray->width,sizeof(int));
if (vetorSomaLinhasVerticais==NULL)
printf("Falha no Malloc vetorSomaLinhasVerticais\n");

vetorDerivadaLinhasVerticais = (int *) calloc ((int)gray->width,sizeof(int));
if (vetorDerivadaLinhasVerticais==NULL)
printf("Falha no Malloc vetorDerivadaLinhasVerticais\n");

vetorDerivada2LinhasVerticais = (int *) calloc ((int)gray->width,sizeof(int));
if (vetorDerivada2LinhasVerticais==NULL)
printf("Falha no Malloc vetorDerivada2LinhasVerticais\n");

//Zera vetores para retirar lixo de memória
zeraVectorMalloc(vetorSomaLinhasVerticais,gray->width);
zeraVectorMalloc(vetorDerivadaLinhasVerticais,gray->width);
zeraVectorMalloc(vetorDerivada2LinhasVerticais,gray->width);

//Calcula Brilho Médio contando apenas pixels diferente de preto puro (valor zero)
vetorMedidas[i].brilhoMedio = vetorMedidas[i].brilhoTotal/contaPixelsDiferenteDeZero(gray);
//Grava no arquivo de registros
fprintf (pFile,"BrilhoMedio %i " ,vetorMedidas[i].brilhoMedio);

//Filtro para retirar vultos
FiltroThreshold(gray,vetorMedidas[i].brilhoMedio,255,0);

//Gera o vetor somatório das colunas verticais
somaValorColunasHorizontais(gray,vetorSomaLinhasVerticais);

//Calcula a 1 derivada
calculaDerivadaVetor(gray,vetorSomaLinhasVerticais,vetorDerivadaLinhasVerticais);
//imprimeVetor(gray,vetorDerivadaLinhasVerticais);

//Detecta o Valor e posição dos máximos e mínimos locais de todo vetor
detectaMaiorMenor(vetorDerivadaLinhasVerticais,Maior,posMaior,Menor,posMenor,0,gray->width,0);

//grava valores para arquivo de registros
vetorMedidas[i].iniPoca = posMaior;
vetorMedidas[i].fimPoca = posMenor;
//Calcula tamanho da poça
vetorMedidas[i].tamPoca = posMenor - posMaior;

//Calcula a 2 derivada
calculaDerivadaVetor(gray,vetorDerivadaLinhasVerticais,vetorDerivada2LinhasVerticais);
//imprimeVetor(gray,vetorDerivada2LinhasVerticais);

//Detecta o Valor e posição dos máximos e mínimos locais em uma parte do vetor
```

```
//Note que a posMaior é o inicio da varredura e posMenor é fim advindos da busca anterior
detectaMaiorMenor(vetorDerivada2LinhasVerticais,Maior,posMaior,Menor,posMenor,posMaior,posMenor,3);

//Calcula posição do arame
vetorMedidas[i].posArame =  posMenor + (posMaior -posMenor)/2;

//Grava posição inicial da poça (em PIXELS)
fprintf (pFile,"IniPoca %i " ,vetorMedidas[i].iniPoca);
//Grava posição final da poça (em PIXELS)
fprintf (pFile,"FimPoca %i ",vetorMedidas[i].fimPoca);
//Grava Tamanho da poça (em PIXELS)
fprintf (pFile,"TamPoca %i " ,vetorMedidas[i].tamPoca);
//GRava posição do Arame (em PIXELS)
fprintf (pFile,"Arame %i \n",vetorMedidas[i].posArame);
}
else //caso a imagem possua um brilho maior
{
fprintf (pFile, "img%i Brilho total: %i\n",i,vetorMedidas[i].brilhoTotal);
}
printf("img%i\n",i);
//Caso deseje visualizar
cvShowImage(wndname, gray);
// on_trackbar(0);
 //Sleep(5000);
}
fclose (pFile);
printf("Fim do processamento pressione qualquer tecla para continuar");
    // Wait for a key stroke; the same function arranges events processing
    cvWaitKey(0);
    cvReleaseImage(&image);
    cvReleaseImage(&gray);
    cvDestroyWindow(wndname);

    return 0;
}
```

I Programa analise de imagens offline 2.0

```
#ifdef _CH_
#pragma package <opencv>
#endif
#include "stdio.h"
#ifndef _EiC
#include "cv.h"
#include "highgui.h"
#endif
#include "math.h"

char wndname[] = "Edge";
char tbarname[] = "Threshold";
int edge_thresh = 1;

int vectorMedioFRame[1200];

IplImage *image = 0, *cedge = 0, *gray = 0, *edge = 0;

#define CAPTURA_IMG_VID_MED_POCA 1
#define CAPTURA_IMG_VID_SOMENTE  2
#define CAPTURA_VID_SOMENTE      3
#define VISUALIZA_SOMENTE        4

//#define TEMPO_DE_AQUISICAO 40000 //Tempo em ms
#define NUMERO_DE_QUADROS 6000   //Número máximo de quadros que poderiam ser amostrados
#define TEMANHO_MAXIMO_JANELA 296 //Tamanho máximo da Janela a ser Capturada

////////////////////////////////////////////////////////////////////////
//Novas Funções OpenCV
////////////////////////////////////////////////////////////////////////
#define THRESHOLD_SCAN_LINES 6
#define LINHA_INICIAL_DETECCAO_ARAME 150
#define POSICAO_MEDIA_ARAME 150
#define REGIAO_HORIZONTAL_POSICAO_INICIAL_ARAME 2
#define REGIAO_VERTICAL_A_PARTIR_DAS_BORDAS_INICIAIS_DO_ARAME 50

#define REGIAO_HORIZONTAL_POSICAO_TOQUE_DO_ARAME 10
#define REGIAO_VERTICAL_A_PARTIR_DAS_BORDAS_DO_ARAME 50

#define IMAGEM_INICIAL 0
#define IMAGEM_FINAL 397

////////////////////////////////////////////////////////////////////////
//Dados de Medição da poça de soldagem
////////////////////////////////////////////////////////////////////////
struct Afericoes {
  int posicaoInicialArame;
  int posicaoFinalArame;
  int posicaoInicialBorda;
  int posicaoFinalBorda;
  int posicaotoquearame;
};

Afericoes vetorAfericoes[NUMERO_DE_QUADROS];

int VetorSomaLinhasVerticias[TEMANHO_MAXIMO_JANELA];
int VetorDiffVerticalPontoMedioArame[TEMANHO_MAXIMO_JANELA];
int VetorDerivada1LinhasVerticais[TEMANHO_MAXIMO_JANELA];
int VetorTempDiffVerticalPontoMedioArame[TEMANHO_MAXIMO_JANELA];

int posicaoMediaArame = 0;
int posicaoDeToqueDoArameNaPoca = 0;

int countQuadros =0;
CvPoint ponto1,ponto2;
CvScalar cor;
////////////////////////////////////////////////////////////////////////
```

```
//Funções Para análise de imagens
//////////////////////////////////////////////////////////////////
void FiltroThreshold(IplImage * imagem,int minimo,int maximo,int valorSpofing)
{
for(int i=0;i<imagem->height;i++) //Altura
{
for(int j=0;j<imagem->width;j++) //Largura
{
if((((uchar*)(imagem->imageData + i*imagem->widthStep))[j] < minimo)||(((uchar*)(imagem->imageData + i*imagem->widthStep))[j] > maximo))
((uchar*)(imagem->imageData + i*imagem->widthStep))[j]= valorSpofing;
}
}
}
//Ok Funciona
void zeraVectorMalloc(int *vetor,int tamanho)
{
for(int i=0;i<tamanho;i++) //Tamanho do vetor
{
vetor[i] = 0;
}

}

//ok Funciona defasada no matlab por 1 ´ja que lá não existe vet[0] mas vet[1]
void somaValorColunasHorizontais(IplImage * imagem,int *vetor,int linhaInicial,int linhaFinal)
{
for(int j=0;j<imagem->width;j++) //Largura
{
for(int i=linhaInicial;i<=linhaFinal;i++) //Altura
{
vetor[j] = vetor[j] + ((uchar*)(imagem->imageData + i*imagem->widthStep))[j];
}
}
}

//ok Funciona defasada no matlab por 1 ´ja que lá não existe vet[0] mas vet[1]
void calculaDerivadaVetor(int tamanhoVetor,int *vetorIn,int *vetorOut)
{
for(int i=0;i<tamanhoVetor-1;i++) //Largura
{
vetorOut[i] = vetorIn[i+1] - vetorIn[i];
}
}


int valorMinimoRegiao(int *vetor,int colunaInicial,int colunaFinal)
{
int menor = 10000; //deve sempre começar com um valor muito alto
int posMenor = 0; //Valor incial em zero
for(int j=colunaInicial;j<colunaFinal;j++)
{
if (vetor[j] < menor)
{
menor = vetor[j];
posMenor = j;
}
}
return posMenor;
}

int valorMaximoRegiao(int *vetor,int colunaInicial,int colunaFinal)
{
int maior = -100000; // deve sempre começar com um valor muito baixo
int posMaior = 0; //Valor incial em zero
for(int j=colunaInicial;j<colunaFinal;j++)
{
if (vetor[j] > maior)
{
maior = vetor[j];
posMaior = j;
}
}
return posMaior;
}

int main( int argc, char** argv )
```

```
{
char str [80];
FILE * pFile;
printf("Analisando Imagens ... \n") ;
for (int i=IMAGEM_INICIAL;i<IMAGEM_FINAL;i++)
{
countQuadros = i;
printf("Quadro %i \n", countQuadros);
(void) sprintf(str,".\\vid\\img%i.bmp", i);
if( (image = cvLoadImage( str, 1)) == 0 )
return -1;

gray = cvCreateImage(cvSize(image->width,image->height), IPL_DEPTH_8U, 1);
cvCvtColor(image, gray, CV_BGR2GRAY);

zeraVectorMalloc(VetorSomaLinhasVerticias,gray->width);
zeraVectorMalloc(VetorDerivada1LinhasVerticais,gray->width);

zeraVectorMalloc(VetorTempDiffVerticalPontoMedioArame,gray->height);
zeraVectorMalloc(VetorDiffVerticalPontoMedioArame,gray->height);

//Filtra ScanLines
FiltroThreshold(gray,THRESHOLD_SCAN_LINES,255,0);

//Filtro de Mediana para suavizar picos
cvSmooth(gray, gray,CV_MEDIAN,3);

//gera o primeiro vetorsoma  para identificar a posição das bordas do arame
somaValorColunasHorizontais(gray,VetorSomaLinhasVerticias,
LINHA_INICIAL_DETECCAO_ARAME-REGIAO_HORIZONTAL_POSICAO_INICIAL_ARAME,
LINHA_INICIAL_DETECCAO_ARAME+REGIAO_HORIZONTAL_POSICAO_INICIAL_ARAME);

//Calcula as derivadas afim de encontrar essas os picos das derivadas
calculaDerivadaVetor(gray->width,VetorSomaLinhasVerticias,VetorDerivada1LinhasVerticais);

//Encontra o inicio da borda do arame encontrando o valor de mínimo restrito a uma regiao
vetorAfericoes[countQuadros].posicaoInicialArame = valorMinimoRegiao(VetorDerivada1LinhasVerticais,
POSICAO_MEDIA_ARAME-REGIAO_VERTICAL_A_PARTIR_DAS_BORDAS_INICIAIS_DO_ARAME,
POSICAO_MEDIA_ARAME+REGIAO_VERTICAL_A_PARTIR_DAS_BORDAS_INICIAIS_DO_ARAME);

//Encontra o Fim da borda do arame encontrando o valor de máximo restrito a uma regiao
vetorAfericoes[countQuadros].posicaoFinalArame = valorMaximoRegiao(VetorDerivada1LinhasVerticais,
POSICAO_MEDIA_ARAME-REGIAO_VERTICAL_A_PARTIR_DAS_BORDAS_INICIAIS_DO_ARAME,
POSICAO_MEDIA_ARAME+REGIAO_VERTICAL_A_PARTIR_DAS_BORDAS_INICIAIS_DO_ARAME);


//Copia a linha vertical do arame para um vetor a partir do ponto medio das bordas do arame
posicaoMediaArame = (vetorAfericoes[countQuadros].posicaoFinalArame -
 vetorAfericoes[countQuadros].posicaoInicialArame)/2 +
 vetorAfericoes[countQuadros].posicaoInicialArame;
//printf("\n O medio valor e: %i\n",posicaoMediaArame);

for(int j=0; j<gray->height;j++)
{
VetorTempDiffVerticalPontoMedioArame[j] = ((uchar*)(gray->imageData + j*gray->widthStep))[posicaoMediaArame];
}


//calcula a derivada da linha vertical afim de encontrar o ponto de toque do arame
calculaDerivadaVetor(gray->height,VetorTempDiffVerticalPontoMedioArame,VetorDiffVerticalPontoMedioArame);

//encontro o valor de máximo ou seja o ponto de toque do arame na poça
vetorAfericoes[countQuadros].posicaotoquearame = valorMaximoRegiao(VetorDiffVerticalPontoMedioArame,0,gray->height);

//gera o segundo vetorsoma  para identificar a posição das bordas do arame
zeraVectorMalloc(VetorSomaLinhasVerticias,gray->width);
zeraVectorMalloc(VetorDerivada1LinhasVerticais,gray->width);

somaValorColunasHorizontais(gray,VetorSomaLinhasVerticias,
vetorAfericoes[countQuadros].posicaotoquearame-REGIAO_HORIZONTAL_POSICAO_TOQUE_DO_ARAME,
vetorAfericoes[countQuadros].posicaotoquearame+REGIAO_HORIZONTAL_POSICAO_TOQUE_DO_ARAME);

//calcula a derivada da linha horizontal afim de encontrar as bordas da poça e arame
calculaDerivadaVetor(gray->width,VetorSomaLinhasVerticias,VetorDerivada1LinhasVerticais);

//Encontra o inicio da borda da poça encontrando o valor de máximo restrito a uma regiao
```

```
vetorAfericoes[countQuadros].posicaoInicialBorda = valorMaximoRegiao(VetorDerivada1LinhasVerticais,
0,vetorAfericoes[countQuadros].posicaoInicialArame - REGIAO_VERTICAL_A_PARTIR_DAS_BORDAS_DO_ARAME);

//Encontra o Fim da borda da poça encontrando o valor de mínimo restrito a uma regiao
vetorAfericoes[countQuadros].posicaoFinalBorda = valorMinimoRegiao(VetorDerivada1LinhasVerticais,
vetorAfericoes[countQuadros].posicaoFinalArame +
REGIAO_VERTICAL_A_PARTIR_DAS_BORDAS_DO_ARAME,gray->width);

////////////////////////////////////////////////////
////////////////////////////////////////////////////
//DEsenha Linha de centro da imagem

ponto1.x =0;
ponto1.y =LINHA_INICIAL_DETECCAO_ARAME;

ponto2.x =gray->width;
ponto2.y =LINHA_INICIAL_DETECCAO_ARAME;


cor.val[0] = 255;
cor.val[1] = 0;
cor.val[2] = 0;
cor.val[3] = 0;

cvLine(gray, ponto1, ponto2, cor,1, 8, 0);

//DEsenha Linhas das bordas do arame e posição media
//desenha Linha da borda Arame inicial

cor.val[0] = 200;
ponto1.x =vetorAfericoes[countQuadros].posicaoInicialArame;
ponto1.y =0;

ponto2.x =vetorAfericoes[countQuadros].posicaoInicialArame;
ponto2.y =gray->height;

cvLine(gray, ponto1, ponto2, cor,1, 4, 0);
//desenha Linha da borda Arame Final
ponto1.x =vetorAfericoes[countQuadros].posicaoFinalArame;
ponto1.y =0;

ponto2.x =vetorAfericoes[countQuadros].posicaoFinalArame;
ponto2.y =gray->height;

cvLine(gray, ponto1, ponto2, cor,1, 4, 0);
//desenha Linha da borda Arame media
ponto1.x =((vetorAfericoes[countQuadros].posicaoFinalArame -
 vetorAfericoes[countQuadros].posicaoInicialArame)/2 +
 vetorAfericoes[countQuadros].posicaoInicialArame);
ponto1.y =0;

ponto2.x =((vetorAfericoes[countQuadros].posicaoFinalArame -
 vetorAfericoes[countQuadros].posicaoInicialArame)/2 +
  vetorAfericoes[countQuadros].posicaoInicialArame);
ponto2.y =gray->height;

cvLine(gray, ponto1, ponto2, cor,1, 4, 0);

//DEsenha Linha d tonhe do Arame na poça
cor.val[0] = 150;
ponto1.x =0;
ponto1.y =vetorAfericoes[countQuadros].posicaotoquearame;

ponto2.x =gray->width;
ponto2.y =vetorAfericoes[countQuadros].posicaotoquearame;

cvLine(gray, ponto1, ponto2, cor,1, 8, 0);

//DEsenha Linhas das bordas Da poça
//desenha Linha da borda  inicial
cor.val[0] = 125;
ponto1.x =vetorAfericoes[countQuadros].posicaoInicialBorda;
ponto1.y =0;

ponto2.x =vetorAfericoes[countQuadros].posicaoInicialBorda;
ponto2.y =gray->height;
```

```
cvLine(gray, ponto1, ponto2, cor,1, 8, 0);
//desenha Linha da borda Arame Final
ponto1.x =vetorAfericoes[countQuadros].posicaoFinalBorda;
ponto1.y =0;

ponto2.x =vetorAfericoes[countQuadros].posicaoFinalBorda;
ponto2.y =gray->height;

cvLine(gray, ponto1, ponto2, cor,1, 8, 0);
//////////////////////////////////////////


(void) sprintf(str,".\\vid2\\img%i.bmp", i);
cvSaveImage(str,gray);

}

pFile = fopen (".\\vid\\AnaliseExperimento.txt","w");

printf("Criando Arquivo de analise ... \n");
for (int i=IMAGEM_INICIAL;i<IMAGEM_FINAL;i++)
{
fprintf (pFile, "img%i ",i);
countQuadros =i;
fprintf (pFile, "posicaoInicialBorda: %i ",vetorAfericoes[countQuadros].posicaoInicialBorda);
fprintf (pFile, "posicaoFinalBorda: %i ",vetorAfericoes[countQuadros].posicaoFinalBorda);
fprintf (pFile, "posicaoInicialArame: %i ",vetorAfericoes[countQuadros].posicaoInicialArame);
fprintf (pFile, "posicaoFinalArame: %i ",vetorAfericoes[countQuadros].posicaoFinalArame);
fprintf (pFile, "posicaoMediaArame: %i ",(vetorAfericoes[countQuadros].posicaoFinalArame -
 vetorAfericoes[countQuadros].posicaoInicialArame)/2 +
 vetorAfericoes[countQuadros].posicaoInicialArame);
fprintf (pFile, "posicaoToquePoca: %i \n",vetorAfericoes[countQuadros].posicaotoquearame);

}
printf("Fechando o Programa ... \n");

fclose (pFile);
cvReleaseImage(&image);
cvReleaseImage(&cedge);
    cvReleaseImage(&gray);
    cvReleaseImage(&edge);
  //  cvDestroyWindow(wndname);

return 0;
}
```