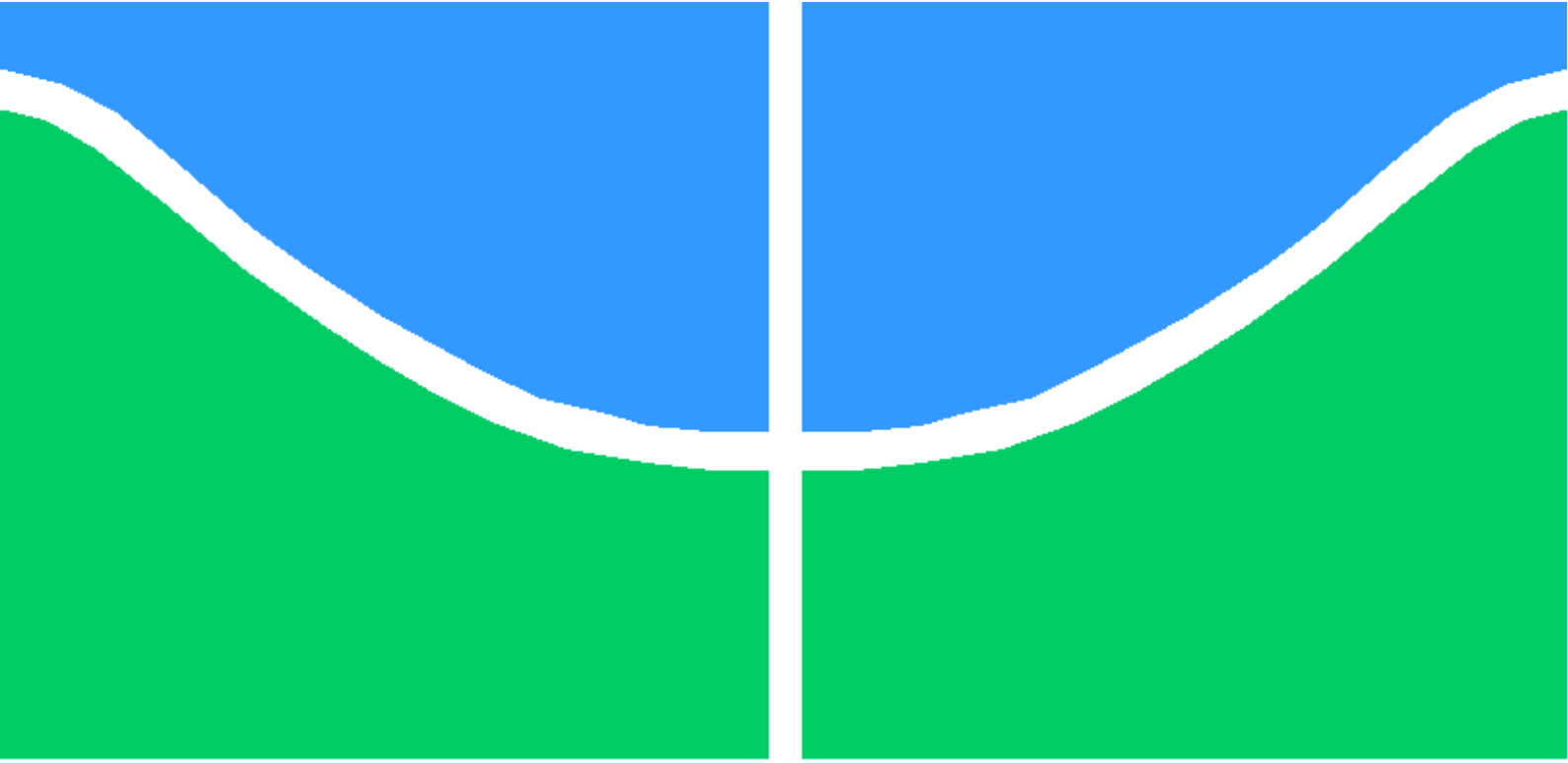


TRABALHO DE GRADUAÇÃO

MONITORAÇÃO NO PROCESSO DE SOLDAGEM GMAW POR MEIO DE VISÃO COMPUTACIONAL COM PROCESSAMENTO VIA FPGA

Rodrigo Ferreira Fernandes

Brasília, Dezembro de 2015

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia

TRABALHO DE GRADUAÇÃO

MONITORAÇÃO NO PROCESSO DE SOLDAGEM GMAW POR MEIO DE VISÃO COMPUTACIONAL COM PROCESSAMENTO VIA FPGA

Rodrigo Ferreira Fernandes

Relatório submetido ao Departamento de Engenharia Mecânica como requisito parcial para obtenção do grau de Engenheiro de Controle e Automação

Banca Examinadora

Prof. Guilherme Caribé, ENM/UnB _____
Orientador

Prof. Magno Batista Corrêa, ENM/UnB _____
Co-orientador

Dedicatória

Dedico este trabalho a...

Rodrigo Ferreira Fernandes

Agradecimentos

A inclusão desta seção de agradecimentos é opcional e fica à critério do(s) autor(es), que caso deseje(em) inclui-la deverá(ao) utilizar este espaço, seguindo está formatação.

Rodrigo Ferreira Fernandes

RESUMO

Continuar o desenvolvimento de um sensor de geometria da poça de fusão em um processo de soldagem GMAW por curto circuito por meio de algoritmos de processamento de imagem que possibilitem para cada quadro:

- Selecionar quadros propícios ao processamento;
- Determinar a largura e o eixo central do arame, bem como a distorção de perspectiva que ocorre no seu plano normal à direção do movimento de soldagem;
- Determinar a largura máxima da poça, assim como a distorção de perspectiva que ocorre no plano da poça de soldagem;

Os resultados do processamento acima serão organizados em vetores, os quais deverão ser filtrados por um Filtro de Kalman para geração de sinais propícios à utilização em uma malha de controle. Os algoritmos deverão ser validados comparando seus resultados com valores medidos à partir de testes controlados do processo.

ABSTRACT

Continue the development of a welding pool geometry sensor in a short circuit GMAW process by means of the development of an image processing algorithm that makes possible at each frame:

- Select frames appropriate to processing;
- Determine the wire's width and central axis, as well as perspective distortion in its plane normal to the welding's movement direction;
- Determine the pool's max width, as well as the distortion perspective in the pool's plane;

The results of the processing above will be organized in vectors, which should be filtered by a Kalman Filter for the generation of signals appropriate for utilization in a control system. The algorithms should be validated comparing its results with measured values from controlled process tests.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

Símbolos Latinos

A	Área	$[m^2]$

Símbolos Gregos

α	Difusividade térmica	$[m^2/s]$

Grupos Adimensionais

Nu	Número de Nusselt

Subscritos

amb	ambiente

Sobrescritos

$\cdot$	Variação temporal

Siglas

ABNT	Associação Brasileira de Normas Técnicas

Capítulo 1

Introdução

1.1 Contextualização

Qualidade de produtos deve ser um foco de atenção da indústria, principalmente em países emergentes, como o Brasil. Isso se deve ao aumento da exigência dos potenciais clientes por produtos e serviços de qualidade [1]. Essa exigência aumenta a competitividade do mercado [2] e obriga as indústrias a aperfeiçoar seus processos produtivos a fim de satisfazer os clientes e manter um bom nível de produtividade a baixo custo. O custo final dos produtos pode ser afetado pelo fator qualidade, aumentando em 80 a elasticidade do preço estimado [3]

Um dos fatores que influencia a qualidade de produtos é o processo de fabricação. A soldagem é um dos processos mais utilizados atualmente na indústria e o processo GMAW (*Gas Metal Arc Welding*) é um dos tipos de solda que apresentam mais vantagens: Alta taxa de deposição, facilidade para automação e variadas aplicações [4, 5].

1.2 Definição do problema

Esse processo requer mão de obra especializada cada vez mais escassa, grandes períodos de trabalho e alta regularidade. Além disso torna o ambiente altamente insalubre com radiação e gases tóxicos proveniente do arco, respingos de metal fundido e altas temperaturas. Este tipo de trabalho faz com que o soldador fique fatigado rapidamente, e isto é uma das principais causas da baixa produtividade e inconsistência em procedimentos com solda manual.

Além disso, regularidade é um aspecto fundamental em soldagem, pois melhora características estruturais e estéticas dos cordões de solda. Um soldador manual não consegue manter parâmetros como velocidade de avanço, altura do arco, ângulo de ataque e posicionamento da pistola por grandes períodos de tempo.

Esses fatores levaram a necessidade automatização para diminuir a exposição de trabalhadores e manter a regularidade do processo, aumentando a produtividade e a qualidade do produto final.

Infelizmente nem mesmo robôs e máquinas específicas para soldagem não são capazes produzir

cordões de solda perfeitos, inclusive podem não atender às especificações desejadas e cometer erros. Portanto, faz-se necessária a aplicação de controle do processo visando minimizar a quantidade de erros, evitar refugo e garantir as características desejadas aos produtos.

Existem vários métodos de controle para soldagem atualmente. A maioria deles consiste em monitorar parâmetros como tensão do arco e corrente, corrigir esses mesmos e outras variáveis como velocidade de alimentação do arame e velocidade de avanço. Esse tipo de controle é de malha aberta pois não monitora o cordão ou a poça de solda e apesar de melhorar a qualidade do processo não garante que os requisitos sejam cumpridos. Para garantir a qualidade da soldagem é necessário que seja feito um controle em malha fechada e isso só pode ser feito ao monitorar os estágios finais do processo.

1.3 Objetivos do projeto

A proposta deste trabalho é esse que o processamento seja feito através de uma FPGA em blocos sequenciais e paralelos. Cada bloco é responsável por uma etapa específica do processamento:

- Seleção de imagens propícias ao processamento;
- Pré-processamento das imagens;
- Obtenção de medidas do eletrodo;
- Obtenção de medidas da poça de soldagem

1.4 Apresentação do manuscrito

No capítulo 2 é feita uma revisão bibliográfica sobre o tema de estudo. Em seguida, o capítulo 3 descreve a metodologia empregada no desenvolvimento do projeto. Resultados experimentais são discutidos no capítulo 4, seguido das conclusões no capítulo 5. Os anexos contém material complementar.

Capítulo 2

Revisão Bibliográfica

2.1 Processo de Soldagem GMAW

O processo de soldagem GMAW do inglês *Gas Metal Arc Welding*, também conhecido como MIG/MAG (*Metal Inert Gas/Metal Active Gas*) é um dos processos de soldagem mais propícios para automação. Alguns aspectos básicos do processo GMAW concernentes ao problema aqui tratado serão detalhados e discutidos nos próximos subitens.

2.1.1 Fundamentos

O princípio da soldagem GMAW é a união de peças metálicas é devido ao aquecimento por um arco elétrico formado entre as mesmas e um eletrodo metálico nu e consumível, sendo todos os elementos envoltos por um gás ativo ou inerte.

O arame de soldar desempenha duas funções: por um lado é o eletrodo que conduz corrente, por outro, é também, em simultâneo, o material de adição a ser inserido na poça de soldagem. Este arame é introduzido mecanicamente através de um alimentador motorizado o que caracteriza o processo como semi-automático quando operado por humanos. [4]

A imagem 2.1 [**?**] demonstra o processo de formação do cordão de solda e seus elementos principais.

Um gás de proteção que flui através do bocal da tocha protege o arco elétrico e o material em fusão, podendo o mesmo ser inerte (MIG) ou ativo (MAG). Os gases inertes, tais como o argônio e o hélio, não entram em reação com o material em fusão e apenas protegem os materiais fundidos de contaminantes. Por outro lado, os gases ativos, não só interferem no próprio arco elétrico, como também reagem com o material em fusão. Um exemplo de gás inativo é uma mistura de dióxido de carbono ou oxigênio com argônio. O componente ativo tem influência, por exemplo, sobre a penetração e/ou a temperatura do banho de fusão.

Além disto, o gás também tem influência nas perdas de elementos químicos, na temperatura da poça de fusão, na sensibilidade à fissuração e na porosidade, bem como na facilidade da execução

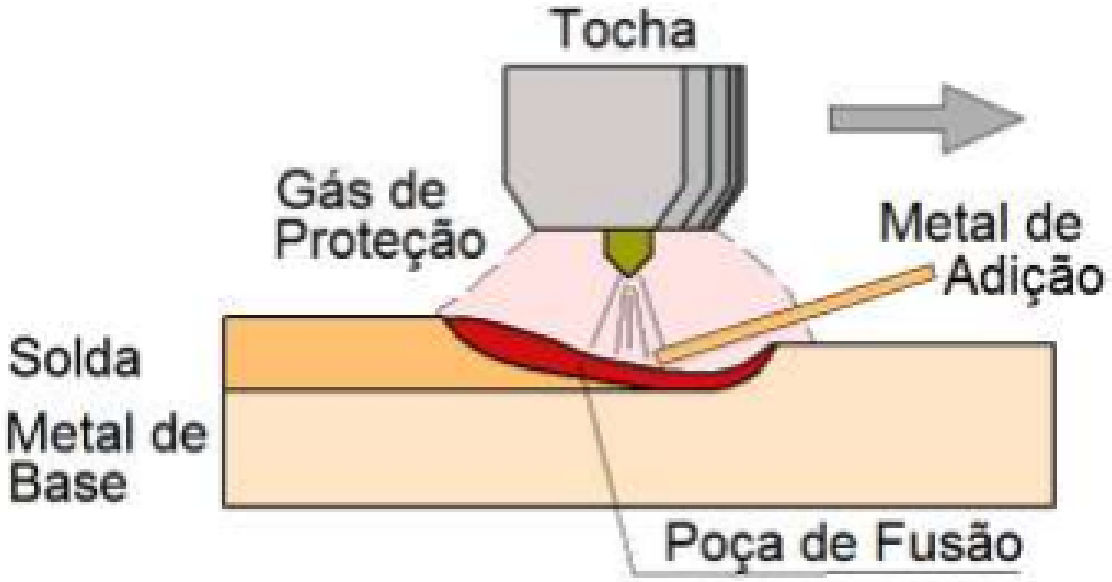


Figura 2.1: Elementos básicos da soldagem GMAW.

da soldagem em diversas posições. Os gases nobres (processo MIG) são preferidos por razões metalúrgicas, enquanto o CO2 puro, é preferido por razões econômicas. O processo MAG é utilizado somente na soldagem de materiais ferrosos, enquanto o processo MIG pode ser usado tanto na soldagem de materiais ferrosos quanto não ferrosos como Alumínio, Cobre, Magnésio, Níquel e suas ligas.

2.1.2 Equipamentos

O processo GMAW tem um conjunto específico de equipamentos. Por ser um processo semi-automático (ou completamente automático), possui mais componentes que métodos mais convencionais de soldagem como o eletrodo revestido e o oxiacetileno.

Os equipamentos necessários para esse tipo de solda são [4]:

- Fonte de energia

  Este processo utiliza sempre corrente contínua. Dependendo do modo em que é feito o controle, pode ser necessário controlar a corrente ou a tensão fornecida pela fonte. É comum usar corrente constante ou tensão constante durante o processo;

- Alimentador de arame

  Existem diversos tipos de alimentadores com diferentes tipos de controle. O controle de alimentação do arame devem estar alinhados com o método de controle elétrico. Por exemplo: é possível usar alimentação constante de arame com tensão constante ou corrente constante e alimentação variável [4];

- Fonte de gás protetor

  Normalmente um cilindro de gás (ou gases) e reguladores de pressão e/ou vazão

- Tocha de soldagem

  Existem diferentes tipos de tocha para proporcionar o desempenho máximo na soldagem para diferentes tipos de aplicações. Elas variam desde tochas para ciclos de trabalho pesados para atividades envolvendo altas correntes até tochas leves para baixas correntes e soldagem fora de posição. Em ambos os casos estão disponíveis tochas refrigeradas a água ou secas (refrigeradas pelo gás de proteção), e tochas com extremidades retas ou curvas.

  A tocha consiste em todo a aparato que é levado até o local de união das peças, seja pela mão de um operador ou uma máquina/robô. Ela contém os seguintes elementos básicos que podem ser vistos na figura 2.2 [6]:

  – Bocal: guia o gás para a junção, geralmente feito de cobre ou material cerâmico. Seu diâmetro deve ser compatível com a corrente de soldagem e o fluxo de gás;

  – Bico de contato: faz o contato elétrico com o eletrodo, feito de cobre;

  – Alavanca de comando ou Gatilho: Ativa a energização do circuito de soldagem, o alimentador de arame e o fluxo de gás.

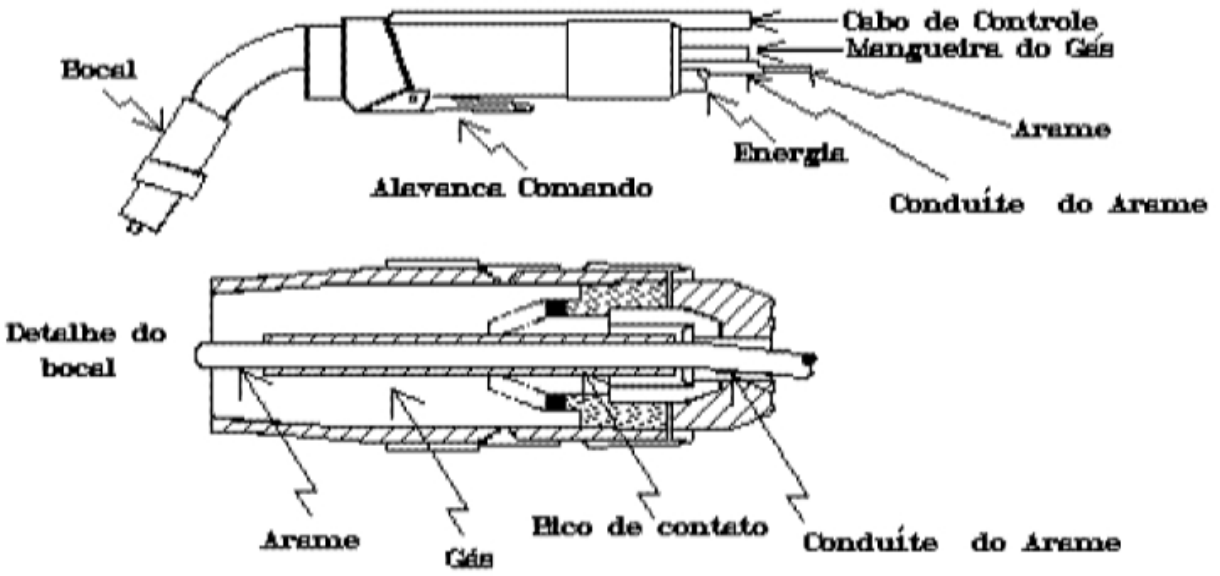


Figura 2.2: Esquemático da tocha da soldagem GMAW.

2.1.3 Monitoramento dos parâmetros de soldagem

Diversas variáveis do processo GMAW podem ser ajustadas para uma boa soldagem do metal. Estas variáveis são a velocidade de alimentação do eletrodo, a distância do bocal à peça, o *stickout*, a inclinação de trabalho do eletrodo, e o fluxo de gás. Essas variáveis requerem um monitoramento

constante por parte do operador, ou por um equipamento automático, que é o escopo deste trabalho. A velocidade de soldagem, a posição de soldagem e o diâmetro do eletrodo também influenciam consideravelmente na geometria do cordão de solda.

2.2 Processamento de imagens

As imagens obtidas no trabalho anterior de Luciano Duarte [7],

As imagens capturadas do processo de soldagem necessitam passar por um processamento para que os valores necessários sejam obtidos. Nesta seção será explicado como é feito esse processamento.

A área de processamento de imagens desperta interesse de estudiosos por ter diversas aplicações em duas principais categorias: (1) aprimoramento de informações pictóricas para interpretação humana; e (2) extração automática de dados relevantes à partir de uma cena [8]. A segunda categoria, que será utilizada neste trabalho, também pode ser designada como "análise de imagens", "visão por computador" ou "reconhecimento de padrões".

Diversas áreas se beneficiam de técnicas de processamento de imagens. Na medicina procedimentos computacionais melhoram o contraste, codificam níveis de intensidade para melhor interpretação e podem até mesmo calcular a quantidade de uma determinada célula em uma imagem. Geógrafos fazem uso de técnicas semelhantes e podem determinar áreas de vegetação, desmatamento, e poluição [9]. Existem muitos outros exemplos, que vão até o controle de qualidade e de processos.

2.2.1 Elementos de sistemas de processamento de imagens

Um sistema de processamento de imagens consiste alguns elementos básicos mostrados a seguir:

1. Aquisição:

   São necessários: um dispositivo físico sensível a uma faixa de frequência no espectro eletromagnético (preferencialmente a luz visível) que produza um sinal elétrico proporcional ao nível de energia detectado; e um digitalizador que converte o sinal elétrico analógico em um sinal digital.

2. Armazenamento:

   Um desafio em processamento de imagens, o armazenamento de imagens digitais requer muita memória RAM ou *frame buffers (Armazenamento por curto tempo) e espaço em disco (Arquivamento).* ***Apesar de os computadores atuais não terem grandes problemas com armazenamento, alguns equipamentos podem não ter memória suficiente para a tarefa como, por exemplo, um microcontrolador.***

3. Processamento:

É um procedimento algorítmico que normalmente é realizado via software. Em casos em que velocidade é um fator importante no processamento pode ser necessário o uso de hardware especializado. É a parte mais complexa do sistema, exige pesquisa e desenvolvimento.

4. Comunicação:

   A transmissão de imagens digitais necessita de uma alta largura de banda. Técnicas de compressão de imagens costumam ser utilizadas para reduzir este problema. Em aplicações em tempo real é necessária a sincronização dos dados.

5. Exibição:

   Por fim, o resultado do processamento é exposto ao ser humano através de um monitor. Pode ser desnecessária a exibição de todas as imagens de um processo que for automático.

O elemento mais importante neste trabalho, Processamento de Imagens,possui alguns passos fundamentais para se chegar ao objetivo proposto. São eles:

1. **Pré-processamento**: tem por objetivo melhorar a qualidade da imagem para que os estágios seguintes tenham mais garantia de sucesso. Geralmente envolve técnicas de realce de contrastes, remoção de ruído e isolamento de regiões.

2. **Segmentação**: é a divisão da imagem pré-processada em partes ou objetos constituintes. Por exemplo, caracteres em uma imagem de texto. É uma das tarefas mais difíceis de se implementar e geralmente é o que define se o processamento vai ter sucesso ou não.

3. **Representação e descrição**: nesse estágio, os dados obtidos pela segmentação em forma de pixels passam a ser representados de forma fronteiras e/ou regiões completas. A partir dessa representação, há a descrição de características quantitativas ou qualitativas, por exemplo, uma concavidade ou um buraco.

4. **Reconhecimento e interpretação**: finalmente, os objetos descritos anteriormente devem ser reconhecidos como algum padrão ou um valor para então se fazer a interpretação de um conjunto de objetos reconhecidos e atribuir um significado a esse conjunto. Por exemplo, uma imagem da palavra "sim" deve conter os objetos reconhecidos como "s","i", "m" e ser interpretada com a palavra "sim".

Todos esses passos são possíveis quando se tem uma **Base de conhecimento** prévia que fica codificada no sistema de processamento de imagens. Com esses conhecimentos sobre o problema em questão e os métodos disponíveis pode-se criar uma solução viável. Por exemplo, quando é conhecida a região da imagem que tem informações de interesses é mais fácil segmentar essa imagem.

2.2.2 Realce de imagens

O objetivo das técnicas de realce é obter uma imagem mais apropriada para uma aplicação específica por meio de técnicas de processamento. As técnicas escolhidas, assim como seus parâmetros

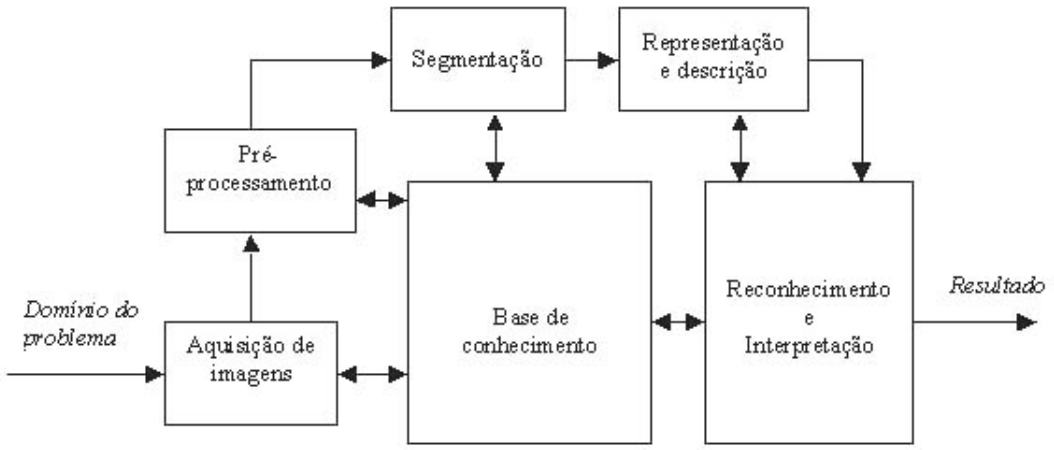


Figura 2.3: Passos Fundamentais em processamento de imagens digitais

e a ordem em que são aplicadas dependem completamente da aplicação. Existem basicamente dois métodos de realce: **Métodos no domínio espacial** e **Métodos no domínio da frequência**.

Os métodos no domínio espacial operam diretamente sobre o agregado de pixels que compõem uma imagem. Funções de processamento de imagens no domínio espacial podem ser expressas como:

$$g(x, y) = T[f(x, y)] \tag{2.1}$$

em que $f(x, y)$ é a imagem de entrada, $g(x, y)$ é a imagem processada e T é um operador sobre f, definido sobre alguma vizinhança de (x, y). T pode operar sobre um conjunto de imagens de entrada, como será explicado mais adiante.

A estratégia é definir uma subimagem em torno de um pixel (x, y) (quadrado, retângulo ou mesmo aproximação de um círculo) e aplicar a operação sobre essa subimagem para cada pixel da imagem e obter g em cada posição correspondente. A vizinhança mais simples que pode ser definida tem tamanho 1×1, e T é uma *trasformação de níveis de cinza* ou *processamtento ponto-a-ponto* da forma:

$$s = T(r) \tag{2.2}$$

em que r e s representam os níveis de cinza de $f(x, y)$ e $g(x, y)$, respectivamente. Este tipo de função permite operações como mudança de contraste, binarização de imagem ou limiarização. Outra variedade de funções pode ser aplicada sobre a imagem ao se trabalhar vizinhanças maiores. Costuma-se utilizar uma janela ou *máscara* em forma de matriz 3×3 para se fazer operações. Essa técnica costuma ser chamada de *filtragem*.

Os métodos no domínio da frequência se baseiam no teorema da convolução com um operador linear invariante com a posição $h(x, y)$ na forma:

$$g(x,y) = h(x,y) * f(x,y) \tag{2.3}$$

ou, a partir do teorema da convolução, no domínio da frequência:

$$G(u,v) = H(u,v)F(u,v) \tag{2.4}$$

em que G, H e F são transformadas de Fourier de g, h e f, respectivamente. Similarmente aos sistemas lineares, a transformada $H(u,v)$ é chamada de *função de transferência óptica*. A equação 2.3 é um processo espacial análogo ao uso de máscaras e $h(x,y)$ costuma ser chamada de *máscara de convolução espacial*.

A seguir são detalhados os métodos de processamento utilizados neste trabalho.

2.2.2.1 Limiarização

Esta é um método espacial que consiste em separar uma imagem em diferentes níveis de cinza. O comum é ter apenas um limiar e dois níveis, o que constitui em uma binarização da imagem. A binarização pode ser feita de duas formas: transformar todos os pixels abaixo de um limiar em zero (preto) ou transfomar todos os acima do limiar no nível máximo (branco).

2.2.3 Distorção de perspectiva

Imagens de objetos tridimensionais capturados por uma câmera podem ter suas dimensões distorcidas pois a imagem é uma projeção bidimensional de parte do objeto. Essa distorção pode tornar impossível a tarefa de medir corretamente dimensões do objeto em questão.

A projeção que ocorre no caso de captura de imagem por uma câmera é do tipo cônica, e não cilíndrica, isso implica que objetos de dimensões iguais no mundo real que estejam a diferentes distâncias da câmera terão suas projeções com diferentes tamanhos na imagem. Além disso câmeras podem ficar em ângulos oblíquos em relação aos planos de interesse dos objetos e essa projeção oblíqua pode resultar em diferentes medidas em diferentes eixos.

Para eliminar o problema de distorção existem dois métodos comumente utilizados: visão estereoscópica e transformação de projeção. O método de visão estereoscópica, apesar de permitir gerar um objeto tridimensional virtual, foi descartado neste trabalho pois seria tecnicamente inviável. Em ambientes com um razoável controle, como manter a câmera fixa em relação ao objeto, a transformação de projeção é suficiente para se obter os dados necessários.

2.2.3.1 Coordenadas Homogêneas

Matematicamente o modelo de câmera e de projeção pode ser definido por uma matriz de transformação H. Pontos do plano real, Π são representados por vetores em letra maiúscula, $\mathbf{X}$,

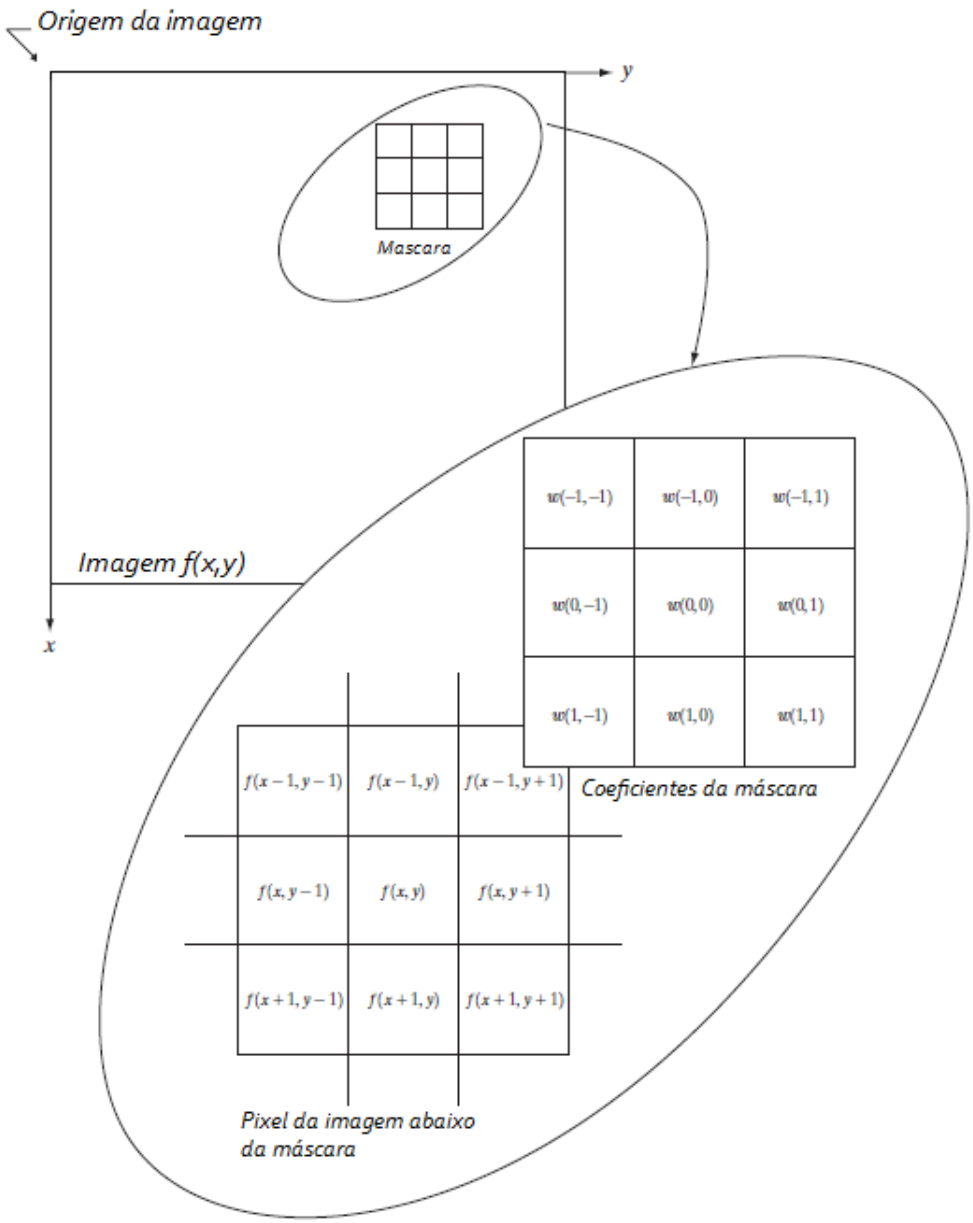


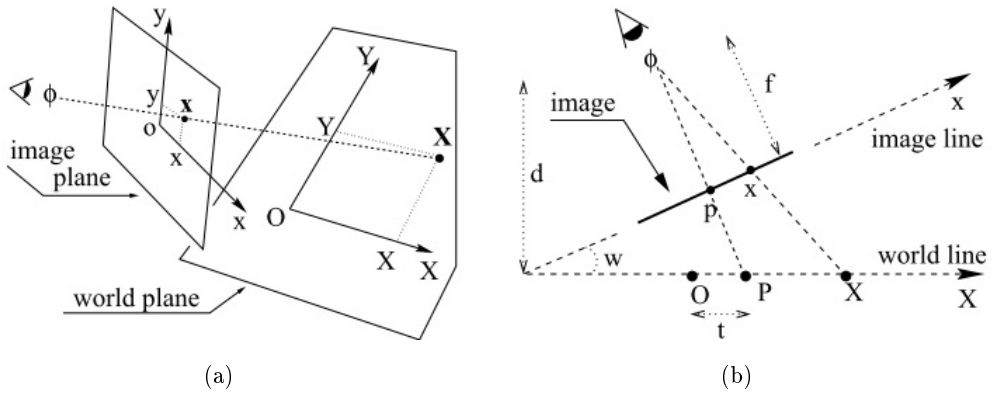
Figura 2.4: Máscara e vizinhança de pixel

e as imagens correspondentes, no plano π são representadas por vetores de letra minúscula, $\mathbf{x}$. A projeção de perspectiva dos pontos correspondentes é dada por [10] :

$$\mathbf{X} = T\mathbf{x} \tag{2.5}$$

Onde T é uma matriz 3×3, "=" representa igualdade em escala. Os vetores de pontos são dados por: $\mathbf{X} = (X_1, X_2, X_3)^T$ e $\mathbf{x} = (x_1, x_2, x_3)^T$

O ponto da imagem, uma projeção, e representado por três coordenadas cartesianas, $\mathbf{x} =$



(a) (b)

Figura 2.5: (a) Modelo de câmera plana. (b) Modelo de câmera unidimensional.

$(x_1, x_2, x_3)^T$. Essas coordenadas são chamadas de homogêneas. Apenas a direção do vetor é importante visto que, independente da distância da câmera, qualquer ponto real em determinada direção aparecerá em um único ponto na projeção. Portanto todos os pontos da forma $\lambda\mathbf{x} = (\lambda x_1, \lambda x_2, \lambda x_3)$ são equivalentes.

Para representar os pontos em um plano cartesiano convencional da forma (x, y), deve-se construir um plano especial π_e), perpendicular ao eixo x_3 a uma distância unitária na direção de x_3. A intersecção do vetor $\mathbf{x}$ com o plano π_e é o ponto $x_e = (x, y, 1)$.

É de interesse que a posição desse plano não afete a posição das coordenadas cartesianas (x, y), portanto define-se essas coordenadas da seguinte forma:

$$x_e = (\frac{x_1}{x_3}, \frac{x_2}{x_3}, 1)^T = (x, y, 1)^T$$

O modelo da câmera é completamente especificado pela matriz T, que pode ser calculada com a posição relativa dos dois planos e o ponto focal da câmera. Porém, essa matriz também pode ser calculada diretamente por correspondência entre pontos na imagem e pontos no mundo real. Esse cálculo é descrito na seção 2.2.3.2.

2.2.3.2 Cálculo da matriz de transformação

A equação 2.2.3.1 pode ser melhor visualizada a seguir:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ t_{31} & t_{32} & t_{33} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix}$$

A escala λ da matriz não afeta a equação, portanto apenas os oito graus de liberdade correspondentes à razão dos elementos da matriz são significantes.

À partir da equação 2.2.3.1,cada correspondência entre pontos reais e pontos da imagem gera duas equações de coordenada cartesianas $\mathbf{H}$. Para n correspondências obtém-se um sistema com

$2n$ equações com 8 variáveis. Se $n = 4$, obtém-se a solução exata [10]. Se $n > 4$, a matriz é super-determinada e estima-se **H** por minimização [11].

A representação em coordenadas cartesianas demonstra a natureza não linear da transformação:

$$x = \frac{x_1}{x_3} = \frac{t_{11}X + t_{12}Y + t_{13}}{t_{31}X + t_{32}Y + t_{33}} \tag{2.6}$$

$$y = \frac{x_2}{x_3} = \frac{t_{21}X + t_{22}Y + t_{23}}{t_{31}X + t_{32}Y + t_{33}} \tag{2.7}$$

Para definir os elementos da matriz de transformação deve-se ter quatro correspondências de pontos entre plano real e projeção. Com a escala de **T** arbitrária, e $t_{33} = 1$, temos os pontos representados por $(\lambda_i x_i, \lambda_i y_i, lambda_i)^T = T(X_i, Y_i, 1)^T$. O sistema de equações lineares resultantes é:

$$\begin{bmatrix} X_1 & Y_1 & 1 & 0 & 0 & 0 & -x_1X_1 & -x_1Y_1 \\ 0 & 0 & 0 & X_1 & Y_1 & 1 & -y_1X_1 & -y_1Y_1 \\ X_2 & Y_2 & 1 & 0 & 0 & 0 & -x_2X_2 & -x_2Y_2 \\ 0 & 0 & 0 & X_2 & Y_2 & 1 & -y_2X_2 & -y_2Y_2 \\ X_3 & Y_3 & 1 & 0 & 0 & 0 & -x_3X_3 & -x_3Y_3 \\ 0 & 0 & 0 & X_3 & Y_3 & 1 & -y_3X_3 & -y_3Y_3 \\ X_4 & Y_4 & 1 & 0 & 0 & 0 & -x_4X_4 & -x_4Y_4 \\ 0 & 0 & 0 & X_4 & Y_4 & 1 & -y_4X_4 & -y_4Y_4 \end{bmatrix} \begin{bmatrix} t_{11} \\ t_{12} \\ t_{13} \\ t_{21} \\ t_{22} \\ t_{23} \\ t_{31} \\ t_{32} \end{bmatrix} = \begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \\ x_3 \\ y_3 \\ x_4 \\ y_4 \end{bmatrix} \tag{2.8}$$

Esse sistema linear garante a solução para a matriz **T**, desde que nenhum grupo de três pontos sejam colineares.

2.2.3.3 Minimização de erros da transformação

Para que se minimize as margens de erro da matriz de transformação **T**, são necessários mais de quatro correspondências de pontos. De modo geral, como demonstrado em [11], quanto mais pontos utilizados na determinação da matriz de transformação, menor é a incerteza. Para n pontos, sistema de equações é semelhante à 2.2.3.2:

$$\begin{bmatrix} X_1 & Y_1 & 1 & 0 & 0 & 0 & -x_1X_1 & -x_1Y_1 \\ 0 & 0 & 0 & X_1 & Y_1 & 1 & -y_1X_1 & -y_1Y_1 \\ X_2 & Y_2 & 1 & 0 & 0 & 0 & -x_2X_2 & -x_2Y_2 \\ 0 & 0 & 0 & X_2 & Y_2 & 1 & -y_2X_2 & -y_2Y_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ X_n & Y_n & 1 & 0 & 0 & 0 & -x_nX_n & -x_nY_n \\ 0 & 0 & 0 & X_n & Y_n & 1 & -y_nX_n & -y_nY_n \end{bmatrix} \begin{bmatrix} t_{11} \\ t_{12} \\ t_{13} \\ t_{21} \\ t_{22} \\ t_{23} \\ t_{31} \\ t_{32} \end{bmatrix} = \begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \\ \vdots \\ x_n \\ y_n \end{bmatrix} \tag{2.9}$$

2.2.4 Detecção de cantos

Muitas tarefas em processamento de imagens requerem a identificação de elementos em imagens, como figuras geométricas, linhas e pontos. Existem diversos métodos e algorítimos para detecção de cantos, mas apenas o método de Harris [12] será descrito nessa seção pois apresenta alta taxa de acerto e fácil implementação de paralelismo, que pode acelerar o processo em um FPGA.

2.2.4.1 Detector de Moravec

O algorítimo parte do princípio de diferenciação intensidade em porções de imagens usado no detector de imagens de Moravec [13]. O detector de Moravec funciona considerando uma janela na imagem, e determina a mudança de intensidade da imagem que resulta do deslocamento dessa janela em várias direções. São feitas as seguintes considerações:

1. Se a imagem na janela é plana, (aproximadamente constante em intensidade), então todas os deslocamentos resultam em uma mudança pequena;
2. Se a janela de move ao longo de uma borda, então a mudança de intensidade é pequena, mas se o deslocamento for perpendicular à borda, a mudança é grande;
3. Se a janela contém um canto ou ponto isolado, então todas os deslocamentos resultam em uma grande mudança de intensidade. Portanto, um canto pode ser detectado quando a menor mudança produzida por qualquer deslocamento for grande.

Foi feita a descrição matemática das considerações acima. Denotando as intensidades como I, a mudança E produzida por um deslocamento (x, y) é dado por:

$$E_{x,y} = \sum_{u}^{v} w_{u,v} |I_{x+u,y+v} - I_{u,v}|^2 \tag{2.10}$$

onde w especifica a janela da imagem: é uma unidade dentro de uma região retangular específica, e zero fora dela. Os deslocamentos (x, y), são do tipo $(1, 0), (1, 1), (0, 1), (-1, 1)$. Portando o detector de Moravec é basicamente: procurar por um máximo local em $min(E)$ acima de uma limite estabelecido.

2.2.4.2 Detector de auto-correlação

A performance do detector de Moravec contém uma série de erros conforme demonstrado em [12], que então os listou e fez as devidas correções:

1. **A resposta é anisotrópica porque apenas um grupo de deslocamentos discretos é considerado** - todos os possíveis deslocamentos podem ser cobertos fazendo uma expansão

analítica em torno do centro de deslocamento.

$$E_{x,y} = \sum_{u}^{v} w_{u,v}[I_{x+u,y+v} - I_{u,v}]^2 = \sum_{u}^{v} w_{u,v}[xX + yY + O(x^2, y^2)]^2 \tag{2.11}$$

onde os primeiros gradientes são aproximados por

$$X = I \otimes (-1, 0, 1) \approx \delta I/\delta x$$
$$Y = I \otimes (-1, 0, 1)^T \approx \delta I/\delta y$$

Então para pequenas mudanças, E pode ser escrito

$$E(x, y) = Ax^2 + 2Cxy + By^2$$

onde

$$A = X^2 \otimes w$$
$$B = Y^2 \otimes w$$
$$C = (XY) \otimes w$$

2. **A resposta é ruidosa porque a janela é binária e retangular** - usar uma janela circular suave, por exemplo uma Gaussiana:

$$w_{u,v} = exp - (u^2 + v^2)/2\sigma^2$$

3. **O operador responde muito cedo a bordas porque o apenas o mínimo de E é considerado** - reformular a medida de canto para usar a variação de E com a direção do deslocamento. A mudança, E, para um pequeno deslocamento (x, y) pode ser escrita como:

$$E(x, y) = (x, y)M(x, y)^T$$

onde a matriz M, 2×2 simétrica é:

$$M = \begin{bmatrix} A & C \\ C & B \end{bmatrix} \tag{2.12}$$

Os autovalores α e β de M correspondem à principais curvaturas da função de autocorrelação local. São três os casos em relação aos autovalores:

1. Se ambos são pequenos, a janela corresponde a uma imagem plana;
2. Se um é pequeno e o outro é grande, a janela corresponde a uma borda;
3. Se ambos são grandes, a janela corresponde a um canto.

Com essas três considerações, se faz necessário uma medida também uma medida de qualidade de cantos. Calcula-se então uma medida R, função de $/alpha$ e $/beta$. Para evitar a decomposição dos autovalores de M, usa-se o traço e a determinante da matriz, sendo que:

$$Tr(M) = \alpha + \beta = A + B$$
$$Det(M) = \alpha\beta = AB - C^2$$

Finalmente, Harris usou a seguinte formulação:

$$R = Det(M) - kTr(M)^2 \tag{2.13}$$

Com essa formulação, pode-se definir o tipo da região localizada em (u, v):

- Canto, se R é positivo;
- Borda, se R é negativo;
- Região plana, se R é pequeno.

Como R pode assumir uma grande gama de valores, positivos e negativos, usa-se um limite inferior e um limite superior

2.3 Sistemas reconfiguráveis

Os custos para se testar circuitos na indústria de eletrônicos são muito elevados, pois é necessário fazer todo o *setup* do processo de produção para um novo circuito. Por esse motivo foi desenvolvida a FPGA, ou *Field Programable Gate Array*, que é um circuito reconfigurável que pode rapidamente assumir as características de um circuito digital, com todas as suas ligações e portas lógicas.

2.3.1 História

Os primeiro dispositivo programáveis foi o *Programmable Read Only Memory*(PROM) , uma memória não volátil que podia ser do tipo fabricado em massa ou programável em campo pelo usuário. Por sua vez o tipo o PROM reconfigurável em campo tinha duas dubdivisões: EPROM (*Erasable* PROM) e EEPROM (*Electronic Erasable* PROM), que pode ser reprogramado múltiplas vezes. Em seguida foram criados os *Programmable Logic Devices* (PLDs), em que o tipo mais comum implementa um conjunto fixo de portas OR precedido por uma matriz de portas AND programáveis. [14]

Em 1985, a empresa Altera criou o primeiro dispositivo reprogramável da indústria, o chip EP300, Os usuários poderiam apagar as células EPROM do EP300 ao emitir luz ultravioleta através de uma janela de quartzo acima do circuito. A opção meramente conveniente à época se mostraria um grande fator na indústria.

As FPGAs foram desenvolvidas como uma evolução dos PLDs, mais especificamente dos PLAs *Programmable Logic Arrays* (PLAs), como descrito em [15]. Em 1985 a empresa Xilinx criou a

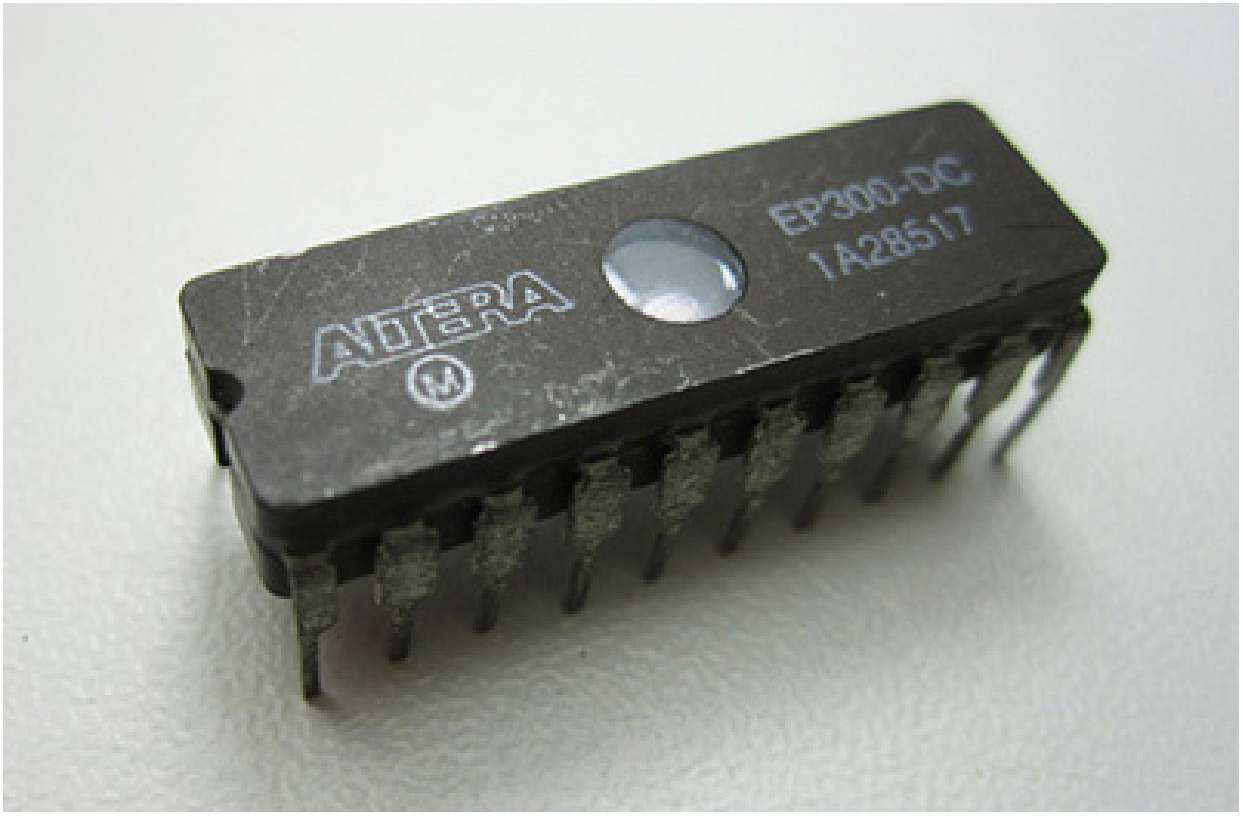


Figura 2.6: EP300 da Altera

primeira FPGA comercial [?], com portas e interconexões reprogramáveis. Contudo, foi durante a década de 1990 que esses dispositivos ficaram famosos e assumiram diversas aplicações.

Atualmente os sistemas reconfiguráveis continuam em alta, principalmente agora que as tecnologias de fabricação estão próximas do limite de miniatuzação. O poder de processamento de arquiteturas convencionais (Arquitetura de von Neumann), baseadas em um *hardware* genérico que recebe fluxos de instruções, demonstra desaceleramento. Com o esperado limite da Lei de Moore , muitos projetistas recorrem ao uso de *hardware* específico para solucionar diferentes problemas.

As FPGAs atuais são produzidas com uma série de facilidades que incluem: memória, entradas e saídas de alta velocidade e blocos lógicos. Elas podem tanto ser utilizadas para desenvolvimento e teste de circuits digitais (motivo pelo qual foram desenvolvidas) como podem ser implementadas no produto final. As aplicações vão desde processamento de imagens a mineração de *bitcoins*

2.4 Aspectos Gerais

De acordo com [15] os *Programmable Gate Array* (PLAs) reconfiguráveis são descritos abaixo: In general, a PLA is a logic circuit which receives a plurality of digital input signals and generates a plurality of digital output signals wherein each of the digital output signals is a programmable sum-of-product combination of the input signals. In conventional PLA?s, one circuit is provided for generating a plurality of terms which are the logical AND of selected input signals; and another

circuit is provided to generate the output signals by selectively ORing the AND terms. A typical PLA may have a total of x input signals, generate a total of y AND terms from the input signals, and generate a total of 2 output signals by selectively ORing the y AND terms.

Outras tecnologias de semicondutores reprogramáveis também foram desenvolvidas, como os *Application Specific Integrated Circuits* (ASICs), onde o dispositivo é desenvolvido apenas para um tipo de aplicação. Ainda há uma subdivisão dentro das FPGAs:

- *One-Time Programmable* (OTP), que funciona como um ASIC depois de programado;
- SRAM-*based*, que pode ser reconfigurado múltiplas vezes e é o tipo mais utilizado.

O FPGA é um arranjo de CLBs (*Configurable Logic Blocks*) e IOBs(*In/Out Blocks*) ligados por chaves de interconexão, também reconfiguráveis. O CLB é a unidade básica do FPGA e consiste de uma matriz de seleção configurável com quatro ou seis entradas, um circuito de seleção (como um multiplexador), e Flip-Flops.

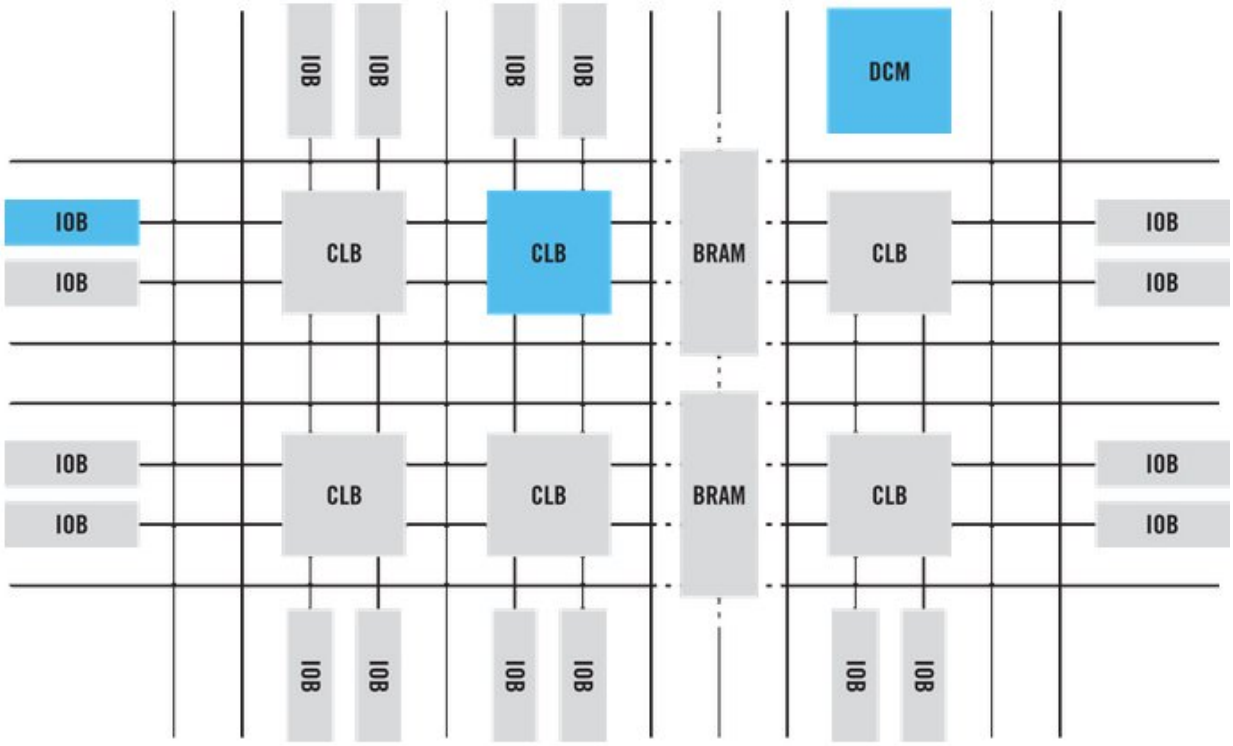


Figura 2.7: Exemplo de esquema de ligação de CLBs e IOBs em uma FPGA

https://en.wikipedia.org/wiki/Field-programmable_gate_array#cite_note-book-16
file:///home/rodrigo/Downloads/US4508977.pdf
https://www.google.com/patents/US8112466?dq=field+programmable+gate+array+patent&hl=pt-BR&sa=X
https://www.google.com/patents/US4870302?dq=freeman+1989+configurable&hl=pt-BR&sa=X&ved=0CBsQ6
https://www.google.com.br/webhp?sourceid=chrome-instant&ion=1&espv=2&ie=UTF-8&client=ubuntu#q=

2.5 Linguagens de Descrição de Hardware e VHDL

Uma linguagem de descrição de hardware, ou HDL (*Hardware Description Language*) é semelhante a uma linguagem de programação em termos de sintaxe e estrutura, porém como o próprio nome diz ela serve mais para descrever do que para programar. Esse tipo de linguagem tem por objetivo facilitar o projeto de circuitos eletrônicos, com uma descrição formal e precisa. Uma HDL padronizada permite a intercambialidade de informações entre diferentes fabricantes e projetistas além da automatização de simulação e síntese de circuitos. Existem variadas HDLs, dentre as quais foi escolhida a VHDL para este trabalho por motivos a serem expostos adiante.

2.5.1 História do VHDL

O surgimento do VHDL se deve à necessidade do *Defense Advanced Research Projects Agency* (DARPA) em desenvolver uma ferramenta de projeto e documentação para o projeto VHSIC (*Very High Speed Integrated Circuit*). O Departamento de Defesa definiu em 1983 os requisitos para uma linguagem de descrição padrão para circuitos e contratou as empresas IBM, Texas e Intermetrics para a tarefa. [16]

A linguagem foi padronizada pelo IEEE (*Institute of Electrical and Electronic Engineers*) com base na versão 7,2 e com o auxílio da empresa CLSI, contratada pela Força Aérea dos Estados Unidos. [16] Em 1987, surgiu o padrão IEEE 1076-1987, denominado VHSIC *Hardware Description Language* ou VHDL. A padronização da linguagem contou com a participação de profissionais de diversas áreas: design de sistemas de computadores, aeroespacial, comunicações, desenvolvedores de CAD, desenvolvimento de circuitos integrados etc. [17]

Em 1993, o padrão 1076 recebeu uma revisão para facilitar ainda mais o trabalho de projetistas. A principal mudança era o melhor gerenciamento hierárquico de arquivos [18]. No mesmo ano surgiu o padrão 1164, que definia o pacote "std_logic_1164" que trouxe mais versatilidade ao modelamento por representar mais condições reais como alta impedância e nós não inicializados. O padrão 1164 é muito utilizado até os dias de hoje, principalmente depois de suas revisão em 2008 .

Em 1997, o surgiu mais um padrão, o 1076.3 [19], que propôs novos tipos de dados como inteiros e reais, e novas funções como soma e multiplicação. Esses tipos são abstratos em código e um mero conjunto de bits no circuito real, porém a linguagem permite as operações apenas com variáveis de mesmo tipo. Essa padronização caracteriza o VHDL como uma linguagem fortemente tipada.

2.5.2 Aspectos gerais da VHDL

A linguagem VHDL, pode ser entendida tanto por máquinas quanto por humanos como na própria definição do IEEE [19]. Tem por objetivo organizar e documentar os projetos para diminuir o tempo de trabalho dos projetistas além de promover a rápida sintetização e de simulação de circuitos.

Uma das suas principais características é a alta hierarquização. É possível criar projetos com múltiplos níveis de hierarquia utilizando entidades (descrições de circuitos) dentro de entidades. Essa estrutura facilita o desenvolvimento partindo de níveis mais altos de abstração para níveis menores, com o método chamado *top down* (muito comum nas linguagens de programação mais recentes). Também é usada a descrição do comportamento esperado do circuito.

Em um circuito sintetizado, a maioria dos comandos é executado concorrentemente, independente do nível hierárquico dos componentes e da ordem em que são declarados. No exemplo da figura 2.8

Apesar disso é possível criar operações sequenciais tais como linguagens de programação em regiões específicas do código, nos subprogramas e processos. Essas estruturas são muito úteis para operações específicas como operações aritméticas mais complexas.

Algumas das características principais da linguagem são:

- *Case-insensitive*, ou seja não identifica diferença entre caixa alta ou baixa em seu código;
- Fortemente tipada;
- Escalonável;
- Hierárquica

2.5.3 Entidades

É uma abstração primária do VHDL. As entidades podem ser instanciadas dentro de outras entidades, formando assim a hierarquia do projeto. Cada instância é vista pela entidade superior como uma caixa preta da qual se conhece apenas as entradas e saídas. A ferramenta de sintetização cria as instâncias à partir do nível mais alto da hierarquia, a Entidade de Projeto (ou *Top Level Entity*).

Cada entidade ser declarada com a palavra reservada "ENTITY" seguida do nome que a identifica. Em seguida podem ser opcionalmente declaradas as cláusulas PORT" ou "GENERIC". A cláusula "PORT" define os tipos de portas de entrada e saída da entidade. A cláusula "GENERIC" permite passar informações estáticas para uma entidade. Essas cláusulas são opcionais e podem ser omitidas, por exemplo em um *testbench* usado para simulações.

O comportamento de uma entidade é definido pela arquitetura e precisa ser declarado com a palavra "ARCHITECTURE" seguida do nome dado a arquitetura e a identificação da entidade a qual ela está definindo. Em seguida, podem ser declarados sinais, constantes, subprogramas ou identificação de outras entidades que serão instanciadas dentro desta. Finalmente, a lógica é descrita entre as palavras "BEGIN" e "END" da arquitetura, com todos os comandos, as operações e instanciações.

O exemplo abaixo mostra uma entidade que utiliza a instância de outras em seu código:

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
```

```
USE ieee.std_logic_arith.all;

ENTITY main is
    port (  CLOCK : in std_logic;
start_stop : in std_logic ;
            up_down      : in std_logic ;
D1A, D1B, D1C, D1D, D1E, D1F, D1G, DP1 : out std_logic;
);


END main ;

ARCHITECTURE ESTRUTURA of main is

SIGNAL BCD : std_logic_vector(3 downto 0);
SIGNAL binario4bits : integer(11 downto 0);
COMPONENT CONTADOR
PORT  (
CLK   : in std_logic;
start_stop   : in std_logic ;
    up_down    : in std_logic ;
binario4bits  : out integer(3 downto 0)
);
END component;

COMPONENT BIN_7SEG
PORT  (
BIN : in std_logic_vector(3 downto 0);
SEG_A, SEG_B, SEG_C, SEG_D, SEG_E, SEG_F, SEG_G, DP : out std_logic
);
END component;

BEGIN
conta1 : CONTADOR port map (CLOCK, start_stop, up_down, binario12bits);
conv1 : BIN_7SEG port map (binario4bits, D1A, D1B, D1C, D1D, D1E, D1F, D1G, DP1);
END ESTRUTURA;
```

2.5.3.1 Entidade de Projeto

A entidade de projeto ou *top level entity* é o nível mais alto na hierarquia de um projeto VHDL. Por convenção, o documento dessa entidade geralmente leva o nome do projeto e é por

este documento que a ferramenta de sintetização começa a leitura.

Assim como outras entidades, a entidade de projeto tem entradas e saídas de diferentes tipos, porém estas não são utilizadas por outras entidades pois esta não está contida em nenhuma outra. As portas declaradas para essa entidade são diretamente relacionadas com o ambiente externo ao dispositivo. Por exemplo: são as portas desta entidade que o sintetizador usa para mapear os pinos de uma placa FPGA. Todas as outras portas de outras entidades são sinais internos à placa neste caso.

A arquitetura da entidade de projeto define o comportamento geral de todo o circuito, com as principais entidades que serão instanciadas e o relacionamento entre as mesmas. Em grandes projetos é comum não haver operações nesta entidade, somente ligações entre outras entidades e entre estas com o meio exterior.

2.5.4 Bibliotecas e Pacotes

O VHDL faz uso de bibliotecas para reaproveitar códigos e padronizar os projetos. Existem bibliotecas padrão como a IEEE, que possui diversos pacotes, como o 1076.3 e o 1164, que geralmente podem ser usados em um mesmo projeto. O projetista também pode definir bibliotecas próprias para fins de organização ou uso em outros projetos.

A maioria das bibliotecas precisam ser declaradas no início do código com a palavra chave "LIBRARY" para se ter acesso a suas definições e pacotes. Há duas bibliotecas que estão sempre disponíveis automaticamente sem necessidade de declaração nos documentos: a biblioteca padrão "std" e a biblioteca "work".

Os pacotes são definições que podem ser utilizadas em diferentes modelos definidos em diferentes arquivos e estão sempre contidos em alguma biblioteca. É um recurso que permite reutilizar código para poupar tempo e principalmente para manter a padronização de projetos. Em uma linguagem fortemente tipada, como é o caso do VHDL, é de suma importância a utilização de pacotes em detrimento de definições locais.

Existem diversos pacotes padrões, muitos dos quais distribuídos pelo IEEE, mas também há os pacotes definidos pelo próprio projetista como citado anteriormente.

Cada pacote utilizado precisa ser declarado no início do documento para se ter acesso aos elementos definidos no mesmo ao longo do código. É necessário declarar de antemão a biblioteca em que o pacote está contido, caso não esteja nas bibliotecas "std" ou "work". O exemplo abaixo permite acesso aos tipos e operações do pacote 1164 da IEEE:

```
LIBRARY ieee;
USE ieee.std_logic_1164;
```

Pacotes definidos pelo usuário precisam ser claramente identificados pela palavra chave "package" e ficam disponíveis para seus documentos dentro da biblioteca "work", a não ser que sejam

explicitamente declaradas dentro de outra biblioteca definida pelo usuário. O exemplo abaixo mostra a declaração e utilização de um pacote:

```
PACKAGE meu_pacote IS
TYPE \textbf{meu_inteiro} IS range -127 to 127;
END meu_pacote;

USE work.meu_pacote.all;

ENTITY minha_entidade IS
PORT(
entrada : in \textbf{meu_inteiro};
saida: out \textbf{meu_inteiro} );
END minha_entidade;
```

2.5.5 Sintetização

Por ser uma linguagem muito abrangente e complexa, algumas das funcionalidades possíveis na linguagem podem não ser possíveis na implementação física, como leitura de arquivos. Portanto é importante considerar o circuito real desejado. A ferramenta de síntese e simulação também deve ser considerada, pois a mesma pode aceitar diversos comandos irrealizável no mundo real.

A sintetização ocorre em várias etapas para levar a descrição do nível de abstração da linguagem até o circuito real. Os fabricantes de dispositivos reprogramáveis (FPGAs) costumam fornecer ferramentas que fazem todo o processo automaticamente. Esses níveis geralmente são:

1. Análise

   Verificação de sintaxe, tipos e hierarquia do código, semelhante a uma primeira passagem de um compilador. A ferramenta de síntese pode também gerar uma descrição VHDL mais simplificada nessa etapa através de iterações, traduzindo macros para estruturas mais simples até obter uma descrição sintetizável.

2. Geração de circuito RTL (*Register Transfer Level*)

   É gerado um circuito de primitivas genéricas no sintetizador, como contadores, somadores e portas lógicas

3. Portas

   As primitivas genéricas são substituídas por componentes reais do dispositivo empregado, como flip-flops, carrys e latches.

4. *Place and Route*

   Com a descrição de componentes e ligações entre os mesmos, o sintetizador faz o posicionamento das primitivas em um dispositivo, utilizando os componentes e ligações disponíveis.

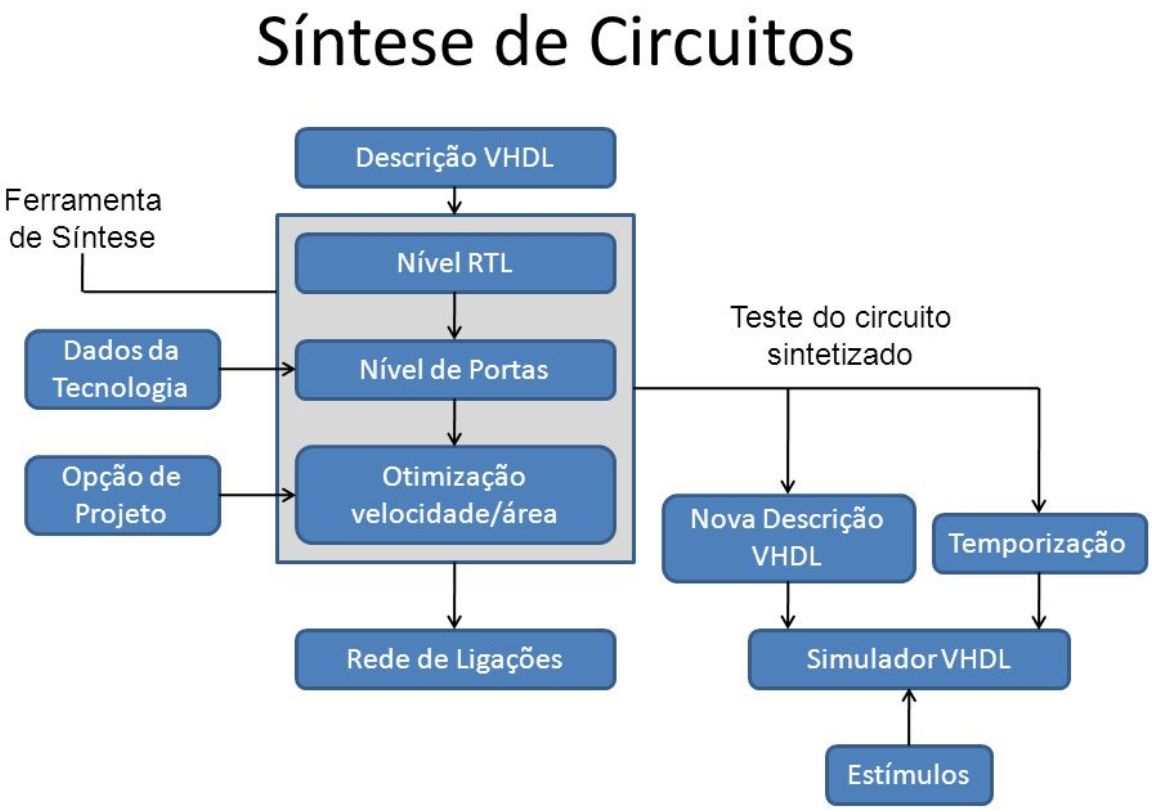


Figura 2.8: Síntese de uma descrição VHDL

2.5.6 Simulação

2.6 Filtro de Kalman

Retirar ruídos dos dados gerados pelo processamento de imagens. Geração do sinal de controle à partir dos vetores gerados pelo algoritmo de processamento.

2.7 VHDL

Capítulo 3

Desenvolvimento

Resumo opcional.

3.1 Introdução

3.2 Arquitetura geral

O sistema devera implementar um controlador em malha fechada, que teoricamente será superior ao controle de malha aberta comumente utilizado em sistemas soldas automatizadas. Ou seja, o sinal de entrada deve ser o resultado da soma do sinal de referência, que será o trajeto definido na programação de um robô, e do sinal de controle, que será calculado por um computador.

Para que o sistema seja realimentado é necessário obter medidas da saída do sistema e, se possível, outros estados. O estado de saída no caso será obtido através de um algoritmo que irá processar fotos do processo e retornar valores numéricos relevantes à operação. Esta etapa é feita com o uso de uma câmera de alta taxa de captura de imagens, capaz de fotografar o arame e a poça de soldagem no instante em que há um curto circuito. O algoritmo processa as imagens e obtém valores numéricos de tamanho da poça, desvio e inclinação do arame. Finalmente, o algoritmo calcula e fornece um sinal de controle para corrigir o posicionamento da tocha de soldagem ou parâmetros como velocidade de alimentação do arame e corrente elétrica.

3.3 Identificação do ângulo da câmera e distorção de perspectiva

A câmera de ser posicionada obliquamente em relação à poça de soldagem não apenas por questões físicas mas também para que inclua tanto poça de soldagem quanto arame. Isso significa que os elementos (poça e arame) ficam em diagonal com o plano da imagem, o que causa uma distorção de perspectiva do que seria a imagem ideal para processamento. Dessa forma, a distorção de perspectiva da imagem deve ser conhecida para que seja possível obter valores corretos no processamento. Uma forma de obter essa informação seria por inserção de dados como ângulo, distância entre câmera e tocha, e distância focal da lente por um operador no sistema. Mas considerando que o sistema possa ser utilizado com câmeras, atuadores e posicionamentos diferentes, inclusive

com a possibilidade de alteração de alguma variável entre dois *setups* diferentes, há a possibilidade de erros e aumento do tempo de *setup*.

Portanto o sistema deve obter esses dados através de uma pré-calibração simples e rápida. Esta pré calibração é feita com o processamento de uma imagem inicial. A imagem deve conter o mínimo de quatro pontos conhecidos, como descrito em 2.2.3.2 no espaço tridimensional para que seja possível calcular a matriz de transformação.

A pré calibração é feita utilizando-se um objeto quadrado de dimensões conhecidas no plano paralelo à peça que será soldada. Isso pode ser feito colocando-se uma pequena chapa metálica quadrada sobre o plano de soldagem. Apenas uma imagem é necessária para se obter a matriz de transformação, desde que o resto do ambiente contido na imagem seja bem controlado no momento da calibração. Os algoritmo gerado com Matlab para gerar a matriz de transformação se mostrou eficiente, mas pode apresentar erros caso haja na imagem cantos mais definidos que os cantos do quadrado.

Esse algoritmo utiliza o método de detecção de cantos de Harris descrito em 2.2.4

Para implementação em uma FPGA, devem-se separar blocos da imagem para serem processados paralelamente. Dessa forma uma imagem separada em e

3.4 Implementação em Matlab

O primeiro algoritmo foi desenvolvido para funcionamento de forma sequencial, pois seria implementado com o software Matlab. A sequência de operações foi definida de acordo com a descrição abaixo. As descrições entre parênteses indicam os parâmetros utilizados em testes que obtiveram bons resultados.

1. Limpar memória e fechar figuras
2. Carregar quantidadeImagens imagens (quantidadeImagens=3)
3. Criar imagem média das imagens carregadas -> I
4. Retirar pixels abaixo de threshold1 para retirar scanlines (threshold1 = 10)
5. Filtro de mediana em I para retirar ruído -> I2
6. Retirar pixels abaixo de threshold2 de I2 -> B (threshold2 = 20)
7. Encontrar topo e base do arame

   Criar vetor de soma do perfil horizontal de B -> somaHor

   Criar vetor de derivadas de somaHor -> derivadaHor

   Encontrar topo do arame

   Valor máximo de derivadaHor entre o topo da imagem e o provável meio do arame (meioVert = 125) -> posArameTopo

Encontrar base do arame

Valor mínimo de derivadaHor entre o meio do arame e o fim da imagem - posArameBase

8. Encontrar borda esquerda e direita da poça de soldagem

   Criar vetor de soma do perfil vertical de B -> somaVert

   Criar vetor de derivadas de somaVert -> derivadaVert

   Encontrar borda esquerda da poça

   Valor máximo de derivadaVert entre o começo e o meio da imagem -> limEsqPoca

   Encontrar borda direita da poça

   Valor mínimo de derivadaVert entre o meio e o final da imagem -> limDirPoca

9. Encontrar laterais do arame

   Criar vetor de derivadas de B na horizontal entre topo e base do arame -> derivadaArame

   Valores mínimos de derivadaArame -> vetor inicioArame

   Valores máximos de derivadaArame -> vetor fimArame

   Linearizar inicioArame e fimArame

Esses passos com os valores para os parâmetros definidos acima obtiveram bons resultados para uma grande quantidade de imagens (imagens antigas). O algoritmo demonstrou certa robustez à ruídos, porém quando a imagem foge muito ao padrão esperado, o algoritmo fornece valores equivocados. Um exemplo é quando há uma grande gota de solda fora da área da poça.

3.5 Implementação do processamento em FPGA

O algoritmo utilizado para implementação em Matlab foi adaptado para uma placa FPGA, utilizando-se a linguagem VHDL. Boa parte dos procedimentos continuou com a característica sequencial, pois as imagens são obtidas desta forma. Alguns procedimentos utilizaram do paralelismo para economia de tempo e recursos. Por exemplo: a obtenção de topo e base do arame e largura da poça de soldagem ocorrem simultaneamente. A seguir é demonstrado como foi feita a implementação de cada item do algoritmo.

3.5.1 Retirada de scanlines

Este filtro de binarização é feito antes de qualquer outro, no momento de recepção de cada pixel. O código para tal é tão simples que foi deixado no corpo da entidade de projeto pois não justifica a criação de uma entidade para tal e não seria adequado colocá-lo em outra.

A binarização consiste basicamente transformar em zero qualquer pixel que seja maior que $threshold1$. O valor utilizado para testes $threshold1$ foi de 10, sendo o brilho máximo 255.

3.5.2 Constantes e tipos customizados

Para agilizar o desenvolvimento do projeto foi criada um pacote com valores e tipos que são recorrentemente utilizados nas diversas entidades. As constantes e tipos dependem basicamente das dimensões das imagens que se pretende processar. São estes:

1. Constantes
    - *numcols*: Largura da imagem, ou quantidade de colunas (pixels) na dimensão horizontal;
    - *numlin*: Altura da imagem, ou quantidade de linhas (pixels) na dimensão vertical.
2. Tipos
    - *vetorHor*: Vetor utilizado para guardar algum tipo de informação de cada linha. Possui *numlin* elementos do tipo inteiro;
    - *vetorVert*: Vetor utilizado para guardar algum tipo de informação de cada coluna. Possui *numcols* elementos do tipo inteiro;
    - *MatrizImagem* Matriz utilizada para armazenar imagens inteiras na simulação. Tem tamanho de *numlin* por *numcols* do tipo *unsigned* de 8b.

3.5.3 Filtro de imagem média e seleção de imagens propícias

Para que fosse implementado este filtro foi utilizado um bloco de memória subdividido em quatro blocos, cada um dimensionado para receber uma imagem. A placa recebe sequencialmente os dados da câmera e armazena três imagens em memória. A imagem média não necessita ser armazenada pois cada pixel pode ser obtido através da média dos pixels das imagens armazenadas à medida que for necessário.

A seleção de imagens propícias ao processamento foi feito no momento de obtenção da imagem. Uma nova imagem é armazenada no bloco marcado como *bloco_atual*. Ao final do recebimento desta, o sinal *bloco_atual* permanece inalterado caso a imagem apresente brilho excessivo ou tem seu valor aumentado de 1 caso contrário. Se *bloco_atual* permanece com o mesmo valor, a imagem mais recente recebida é sobrescrita pela nova imagem que será recebida em seguida. Se *bloco_atual* for acrescido, a imagem mais antiga é sobrescrita. Desta forma, à partir da terceira imagem sempre há três imagens úteis carregadas e uma sendo recebida.

Para que isto fosse possível durante a simulação, foi implementado um bloco de RAM utilizando uma *megafunction* do Quartus II. O tamanho desta memória calculado para armazenar quatro vezes a quantidade de bits de uma imagem. Para os primeiros estágios do desenvolvimento e teste, foram utilizadas imagens de 296x264 pixels reduzidas a 60x60 pixels (para economia de tempo e melhor visualização da simulação). O tamanho desta memória deveria comportar $60 \times 60 \times 4 = 1440B$, portanto foi utilizada uma memória de 16 KB.

O interessante dessa implementação é que o endereçamento de leitura e escrita pode utilizar diretamente o sinal *bloco_atual* com uma simples concatenação, o que poupa operações na implementação em VHDL e recursos na placa.

Como será visto mais detalhadamente abaixo em 3.5.4, os cálculos e operações são feitos a cada novo pixel durante o carregamento de uma imagem. Assim, a média de cada pixel das últimas três imagens válidas deve ser calculada a cada ciclo de *clock*. Como os valor absoluto não são de interesse durante os cálculos ficou decidido que a soma das três imagens seria suficiente, o que evita uma divisão desnecessária.

Durante a implementação foi constatado que fazer a leitura de diversos pixels em diferentes endereços de memória podeira consumir muito tempo e talvez se tornar proibitivo caso se deseje uma quantidade maior de imagens no filtro de média. A solução para este problema foi utilizar uma espécie de média móvel nos sinais que são utilizados para os cálculos. Em vez de se fazer a leitura de dois endereços de memória mais recentes para somá-los com o novo pixel recebido (*dado_escrita*), foi feita apenas a leitura do mais antigo par ser subtraído da soma. Por exemplo: quando era feito o recebimento de um pixel da linha *linha*, coluna *coluna* o valor do elemento $somaHor(linha)$ era atualizado da seguinte forma:

$$somaHor(linha) = somaHor(linha) + dado\_escrita - q(bloco\_atual - 3, linha, coluna)$$

Onde q é o byte armazenado no endereço de memória dado por *bloco_atual* concatenado com $linha \times numcols + coluna$, sendo *numcols* a largura da imagem.

3.5.4 Cálculo de topo e base do arame

Os dados de topo e base do arame foram obtidos com a mesma lógica criada para Matlab e de forma sequencial. Foi utilizado um processo que tem in_janela e in_clock como *sensitivity list*. O sinal in_janela foi utilizado para sincronização de final de imagem e o in_clock era o *clock* de entrada da câmera e define o fim e o começo da transmissão de cada pixel.

Foram criados os mesmos vetores *somaHor* e o *derivadaHor* ambos do tipo *vetorHor*. Como citado em 3.5.3 os elementos de *somaHor* foram calculados a cada pixel recebido, mas os valores de *derivadaHor* apenas foram calculados ao final da leitura de cada linha.

Ao mesmo tempo foram definidos os valores *posArameTopo* e *posArameBase* com simples testes ao final de cada linha. Ao final da imagem estes valores já estão devidamente definidos.

Capítulo 4

Resultados Experimentais

Resumo opcional.

4.1 Introdução

Na introdução deverá ser feita uma descrição geral dos experimentos realizados.

Para cada experimentação apresentada, descrever as condições de experimentação (e.g., instrumentos, ligações específicas, configurações dos programas), os resultados obtidos na forma de tabelas, curvas ou gráficos. Por fim, tão importante quando ter os resultados é a análise que se faz deles. Quando os resultados obtidos não forem como esperados, procurar justificar e/ou propor alteração na teoria de forma a justificá-los.

4.2 Avaliação do algoritmo de resolução da equação algébrica de Riccati

O algoritmo proposto para solução da equação algébrica de Riccati foi avaliado em diferentes máquinas. Os tempos de execução são mostrados na Tabela 4.1. Nesta tabela, os algoritmos propostos receberam a denominação CH para Chandrasekhar e $CH + LYAP$ para Chandrasekhar com Lyapunov. As implementações foram feitas em linguagem *script* MATLAB.

Observa-se que o algoritmo $CH + LYAP$ apresenta tempos de execução superiores com relação ao algoritmo CH. Entretanto, era esperado que o algoritmo CH fosse mais rápido. Este

Tabela 4.1: Tempos de execução em segundos para diferentes máquinas

Algoritmo	**Laptop 1.8 GHz**	**Desktop PIII 850 MHz**	**Desktop MMX 233**	**Laptop 600 MHz**
Matlab ARE	649,96	1.857,5	7.450,5	9.063,9
CH	259,44	606,4	2.436,5	2.588,5
$CH + LYAP$	357,86	952,9	3.689,2	3.875,0

resultado se justifica pelo fato de o algoritmo CH fazer uso de funções embutidas do MATLAB. Já o aloritmo $CH + LYAP$ faz uso também de funções *script* externas, aumentando bastante seu tempo computacional.

Capítulo 5

Conclusões

Este capítulo é em geral formado por: um breve resumo do que foi apresentado, conclusões mais pertinentes e propostas de trabalhos futuros.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] VENTURA, R. MudanÃğas no perfil do consumo no brasil. August 2010.

[2] BROERING, C. E. *Desenvolvimento de Sistemas para a AutomaÃğÃčo da Soldagem e corte TÃľrmico*. Dissertação (Mestrado) — Universidade Federal de Santa Catarina, 2005.

[3] TRANDEL, G. A. The bias due to omitting quality when estimating automobile demand. *The Review of Economics and Statistics*, August 1991.

[4] MARQUES, P. V.; MODENESI, P. J.; BRACARENSE, A. Q. *Soldagem: Fundamentos e Tecnologia*. [S.l.]: Editora UFMG, 2005.

[5] ADOLFSSON, S. On-line quality monitoring in short-circuit gas metal arc welding. *Welding Journal*, December 1997.

[6] MACHADO, I. G. *Soldagem e tÃľcnicas conexas*. [S.l.: s.n.], 2007.

[7] FRANCO, L. D. N. *SincronizaÃğÃčo, captura e anÃąlise de imagens da poÃğa de soldagem no processo GMAW convencional, no modo de transferÃłncia metÃąlica por curto circuito*. Dissertação (Mestrado) — Universidade de BrasÃŋlia, 2007.

[8] FILHO, O. M.; NETO, H. V. *Processamento Digital de Imagens*. Rio de Janeiro: Brasport, 1999.

[9] GONZALEZ, R. C.; WOODS, R. E. *Processamento de Imagens Digitais*. 1Âł. ed. [S.l.]: Edgard BlÃijcher LTDA, 2000.

[10] MUNDY JOSEPH. ZISSERMAN, A. Geometric invariance in computer vision. In: _____. [S.l.]: MIT Press, 1992. cap. 23.

[11] CRIMINISI, A.; REID, I.; ZISSERMAN, A. A plane measuring device. *British Machine Vision Conferencel*, July 1997.

[12] HARRIS, M. S. C. A combined corner and edge detector. *Plessey Research Roke Manor*, 1998.

[13] MORAVEC, H. P. Towards automatic visual obstacle avoidance. *Proc. 5th International Joint Conference on Artificial Intelligence*, p. 584, 1977.

[14] HISTORY of FPGAs. Disponível em: <url>.

[15] D.W. Page e L.V.R. Peterson. *Re-programmable PLA*. abr. 2 1985. US Patent 4,508,977. Disponível em: <http://www.google.com/patents/US4508977>.

[16] D'AMORE, R. *VHDL: descriçao e síntese de circuitos digitais*. LTC, 2005. ISBN 9788521614524. Disponível em: <https://books.google.com.br/books?id=6I26AAAACAAJ>.

[17] IEEE Standard VHDL Language Reference Manual. *IEEE Std 1076-1987*, p. 01–, 1988.

[18] IEEE Standard VHDL Language Reference Manual. *ANSI/IEEE Std 1076-1993*, p. i–, 1994.

[19] IEEE Standard VHDL Synthesis Packages. *IEEE Std 1076.3-1997*, p. 1–52, June 1997.

ANEXOS

I. DIAGRAMAS ESQUEMÁTICOS

II. DESCRIÇÃO DO CONTEÚDO DO CD